Formalisms for Representing Ontologies: State of the Art Survey

Deliverable TONES-D06

F. Baader[4], D. Calvanese[1], G. De Giacomo[2], P. Fillottrani[1], E. Franconi[1], B. Cuenca Grau[3], I. Horrocks[3], A. Kaplunova[5], D. Lembo[2], M. Lenzerini[2], C. Lutz[4], R. Möller[5], B. Parsia[3], P. Patel-Schneider[6], R. Rosati[2], B. Suntisrivaraporn[4], S. Tessaris[1]

[1] Free University of Bozen-Bolzano, [2] Università di Roma "La Sapienza"
[3] The University of Manchester, [4] Technische Universität Dresden
[5] Technische Universität Hamburg-Harburg, [6] TONES Industrial Advisory Team

Project:	FP6-7603 – Thinking ONtolgiES (TONES)
Workpackage:	WP2– Definition of common logical framework for representing ontologies
Lead Participant:	UOR: University of Rome "La Sapienza"
Reviewer:	Ulrike Sattler
Document Type:	Deliverable
Classification:	Public
Distribution:	TONES Consortium
Status:	Draft
Document file:	D06_FormalismsSurvey.pdf
Version:	2.0
Date:	31 May, 2006
Number of pages:	91

Abstract

In this document we provide a structured overview of formalism for the representation of ontologies developed in Logic and Artificial Intelligence, and survey the state of the art in methods and techniques for automated reasoning studied in Computational Logic. Since a more general overview of such formalisms has already been reported as part of the deliverable D01 "State of the art survey", here we concentrate on a wide family of logics, called Description Logics (DLs). DLs have been developed over the years in Artificial Intelligence and Computational Logic to represent formally knowledge about a domain of interest in terms of objects grouped into classes and relationships between classes. Such formalisms have been often advocated as the formal foundation of ontologies. Indeed current ontology language standards such as RDF/RDFS and especially OWL are based on such formalisms. In this document, we review DLs from several points of view, laying the foundation of the research that will be developed within the TONES Project.

Document Change Record		
Version	**Date**	**Reason for Change**
v.1.1	May 10, 2006	First draft
v.1.2	May 19, 2006	First internal release
	May 22, 2006	Comments by reviewer
v.2.0	May 31, 2006	Final version (addressing comments by reviewer)

Contents

1 Introduction

In this document we provide a structured overview of formalism for the representation of ontologies developed in Logic and Artificial Intelligence, and survey the state of the art in methods and techniques for automated reasoning studied in Computational Logic.

In fact our overview is going to be rather technical, since a more general overview to such formalisms has already been reported as part of the deliverable D1 "State of the art survey". Here we concentrate to a wide family of logics, called Description Logics (DLs) [BCM+03], that have been developed over the years in Artificial Intelligence and Computational Logic to represent formally knowledge about a domain of interest in terms of objects grouped into classes and relationships between classes. Such formalisms have been often advocated as the formal foundation of ontologies. Indeed current ontology language standards such as RDF/RDFS and especially OWL are based on such formalisms.

We review DLs from several point of view in this documents, laying the foundation of the research that will be developed within TONES Project.

We start by concentrating on intensional reasoning, and in Section 2, we review the fundamental computational characterization of the most expressive DLs studied in literature. Such DLs are based on the correspondence between highly expressive description languages and Modal Logics of Programs, in particular Propositional Dynamic Logic (PDL) and its variants. We review the correspondence between DLs and PDLs in details showing how results in one of the two areas can be immediately imported in the other one. Such a correspondence was at the base of the expressiveness leap that DLs made in the 90's, moving from logics for describing structural properties of objects to logics that where able to formally capture conceptual models such as Entity Relationship Diagrams in Databases and UML Class Diagrams in Software Engineering , and later full-fledged ontologies as well. Such logics allow for reasoning wrt fully general DLs assertions (inclusion assertions) that are needed when modeling complex knowledge as in ontologies. Also many of them include forms of fixpoint construct in the language such as transitive closure of least fixpoint solutions. The section includes reference to the main results on reasoning in expressive DLs, and it reports on the boundary between decidability and undecidability, discussing language constructs combination that lead to undecidability.

Among the various DLs mentioned in the previous section there is a family of DLs that has a particular importance: the so call $\mathcal{SH}$ family. Such a family is reviewed in Section 3. The $\mathcal{SH}$ family keeps the ability of expressing complex intensional knowledge by giving up fixpoints, including transitive closure, in favor of weaker transitive roles. This simplification allow for basing reasoning on tableaux that can be highly optimized with the inclusion of intelligent and sophisticated heuristics. Indeed such logics are at the base of all modern automated reasoning systems for expressive DLs such as Fact, Racer, Pellet, etc. In the section also concrete domains are looked up, as well as form of extensional reasoning that have been developed for such logics, such has instance retrieval.

In Section 4, we move to a different kind of DLs, the tractable DLs, which can express intensional knowledge as required in representing ontology, but with suitable limitations so has to make reasoning tractable. In particular we look at standard DLs reasoning tasks, but also at more complex tasks such as query answering for queries expressed in query languages richer than DLs. Indeed, while DLs are quite well suited to formalize conceptual

knowledge as that needed for ontologies, they are quite poor as query languages due to their inability of using variables to construct query complex patterns (notice that the absence of variables is more than welcome when modeling conceptual models, instead). So for query language constructs, the DL community has looked at the work on queries in Databases. In particular, we review current results on answering conjunctive queries in DLs. Observe that, when it becomes of concern to perform data access through ontologies, the computational characterization we are after needs refinement: we need to distinguish between complexity wrt the extensional knowledge (data complexity) and complexity wrt the intensional knowledge. While exponential bounds can be acceptable wrt latter, it is much more critic to accept them for the former, since typically we will have a large quantity of data to access analogously to what happens in databases. So the section looks at reasoning from the data complexity point of view as well.

In Section 5 and 6, we look at forms of reasoning that go beyond classical logical reasoning, that is, forms of reasoning that still have a well specified formal semantics, which make use classical logical reasoning as part of a more complex computation. In particular in Section 5, we review the work done on defining rules over a DL knowledge base, and use them for several forms of computations. In Section 6, we look at various forms of on non-standard inference, such as finding the most specific concept an object is an instance of, or finding the most general subsumer of two concepts, i.e., the "minimal" concept that subsume both of them.

Section 7 ends the document by reviewing current ontology standard languages. In particular, we review OWL (version 1.1) and its fragments and extension, including OWL-DL, OWL-lite, and also RDF and RDFS. The relationships between such languages and DLs presented in the previous sections is spelled out, as well as the forms of reasoning currently implemented in the reasoning systems for them.

In the following we assume the reader to have already some basic knowledge on DLs, which can be acquired, e.g., by looking at the first three chapters of [BCM$^+$03].

2 Expressive Description Logics

Description logics have been introduced with the goal of providing a formal reconstruction of frame systems and semantic networks. Initially, the research has concentrated on subsumption of concept expressions. However, for certain applications, and in particular when representing ontologies, it turns out that it is necessary to assert knowledge by means of inclusion axioms of the most general form (without limitation on cycles) in the TBox. Therefore, there has been a strong interest in the problem of reasoning over knowledge bases of such a general form [BCM$^+$03].

When reasoning over general knowledge bases, it is not possible to gain tractability by limiting the expressive power of the description logic, because the power of arbitrary inclusion axioms in the TBox alone leads to high complexity in the inference mechanisms. Indeed, logical implication is ExpTime-hard even for the very simple language $\mathcal{AL}$ [BCM$^+$03] or even $\mathcal{FL}_0$ [BBL05]. This has lead to investigating very powerful languages for expressing concepts and roles, for which the property of interest is no longer tractability of reasoning, but rather decidability. Such logics, called here *expressive de-*

scription logics, have the following characteristics:

1. The language used for building concepts and roles comprises all classical concept forming constructs, plus several role forming constructs such as inverse roles, and reflexive-transitive closure.

2. No restriction is posed on the axioms in the TBox.

The goal of this section is to provide an overview on the results and techniques for reasoning in expressive description logics.

2.1 Correspondence between Description Logics and Propositional Dynamic Logics

Here, we focus on expressive description logics that, besides the standard $\mathcal{ALC}$ constructs, include regular expression over roles and possibly inverse roles [Baa91, Sch91]. It turns out that such description logics correspond directly to Propositional Dynamic Logics, which are modal logics used to express properties of programs. We first introduce syntax and semantics of the description logics we consider, then introduce Propositional Dynamic Logics, and finally discuss the correspondence between the two formalisms.

2.2 Description Logics

We consider the description logic $\mathcal{ALCI}_{reg}$, in which concepts and roles are formed according to the following syntax:

$$\begin{array}{lcl} C, C' & \longrightarrow & A \mid \neg C \mid C \sqcap C' \mid C \sqcup C' \mid \forall R.C \mid \exists R.C \\ R, R' & \longrightarrow & P \mid R \sqcup R' \mid R \circ R' \mid R^* \mid id(C) \mid R^- \end{array}$$

where A and P denote respectively atomic concepts and atomic roles, and C and R denote respectively arbitrary concepts and roles.

In addition to the usual concept forming constructs, $\mathcal{ALCI}_{reg}$ provides constructs to form regular expressions over roles. Such constructs include *role union*, *role composition*, *reflexive-transitive closure*, and *role identity*. Their meaning is straightforward, except for role identity $id(C)$ which, given a concept C, allows one to build a role which connects each instance of C to itself. As we shall see below, there is a tight correspondence between these constructs and the operators on programs in Propositional Dynamic Logics. The presence in the language of the constructs for regular expressions is specified by the subscript "*reg*" in the name.

$\mathcal{ALCI}_{reg}$ includes also the *inverse role* construct, which allows one to denote the inverse of a given relation. One can, for example, state with $\exists \mathsf{child}^-.\mathsf{Doctor}$ that someone has a parent who is a doctor, by making use of the inverse of role child. It is worth noticing that, in a language without inverse of roles, in order to express such a constraint one must use two distinct roles (e.g., child and parent) that cannot be put in the proper relation to each other. We use the letter $\mathcal{I}$ in the name to specify the presence of inverse roles in a description logic; by dropping inverse roles from $\mathcal{ALC}_{reg}$, we obtain the description logic $\mathcal{ALC}_{reg}$.

From the semantic point of view, given an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, concepts are interpreted as subsets of the domain $\Delta^{\mathcal{I}}$, and roles as binary relations over $\Delta^{\mathcal{I}}$, as follows[1]:

$$\begin{array}{rcl}
A^{\mathcal{I}} & \subseteq & \Delta^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C \sqcap C')^{\mathcal{I}} & = & C^{\mathcal{I}} \cap C'^{\mathcal{I}} \\
(C_1 \sqcup C_2)^{\mathcal{I}} & = & C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} & = & \{o \in \Delta^{\mathcal{I}} \mid \forall o'.\, (o, o') \in R^{\mathcal{I}} \supset o' \in C^{\mathcal{I}}\} \\
(\exists R.C)^{\mathcal{I}} & = & \{o \in \Delta^{\mathcal{I}} \mid \exists o'.\, (o, o') \in R^{\mathcal{I}} \wedge o' \in C^{\mathcal{I}}\} \\
P^{\mathcal{I}} & \subseteq & \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \\
(R \sqcup R')^{\mathcal{I}} & = & R^{\mathcal{I}} \cup R'^{\mathcal{I}} \\
(R \circ R')^{\mathcal{I}} & = & R^{\mathcal{I}} \circ R'^{\mathcal{I}} \\
(R^*)^{\mathcal{I}} & = & (R^{\mathcal{I}})^* \\
id(C)^{\mathcal{I}} & = & \{(o, o) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid o \in C^{\mathcal{I}}\} \\
(R^-)^{\mathcal{I}} & = & \{(o, o') \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (o', o) \in R^{\mathcal{I}}\}
\end{array}$$

We consider the most general form of TBoxes constituted by general inclusion axioms of the form $C \sqsubseteq C'$, without any restriction on cycles. We use $C \equiv C'$ as an abbreviation for the pair of axioms $C \sqsubseteq C'$ and $C' \sqsubseteq C$. We adopt the usual descriptive semantics for TBoxes. An interpretation $\mathcal{I}$ is a *model* of an inclusion assertion $C \sqsubseteq C'$, if $(C)^{\mathcal{I}} \subseteq (C')^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is a *model* of a TBox $\mathcal{T}$ if $\mathcal{I}$ is a model of every assertion in $\mathcal{T}$. A *TBox is satisfiable* if it admits a model. A concept C is *satisfiable wrt a TBox* $\mathcal{T}$ if there exist a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}} \neq \emptyset$. Also we say that a concept C is *satisfiable* if it is satisfiable in an empty TBox. A TBox T *logically implies* an assertion $C \sqsubseteq C'$ if every model $\mathcal{I}$ of $\mathcal{T}$ is also a model of $C \sqsubseteq C'$, if $(C)^{\mathcal{I}} \subseteq (C')^{\mathcal{I}}$.

It is well known that for most DLs (an in particular for all DLs mentioned in this section), TBox satisfiability, concept satisfiability wrt a TBox, and logical implication are mutually reducible to each other (in PTime or less), so in the following we will concentrate on logical implication only. Instead concept satisfiability (in an empty TBox) is typically much simpler than logical implication. We will see however that for most of the DLs considered here this is not the case since one can polynomially reduce logical implication to concept satisfiability via the so call internalization (see later.)

Example 2.1 The following $\mathcal{ALCI}_{reg}$ TBox $\mathcal{T}_{file}$ models a file-system constituted by file-system elements (FSelem), each of which is either a Directory or a File. Each FSelem has a name, a Directory may have children while a File may not, and Root is a special directory

[1] We use $\mathcal{R}^*$ to denote the reflexive-transitive closure of the binary relation $\mathcal{R}$, and $\mathcal{R}_1 \circ \mathcal{R}_2$ to denote the chaining of the binary relations $\mathcal{R}_1$ and $\mathcal{R}_2$.

which has no parent. The parent relationship is modeled through the inverse of role child.

$$\begin{aligned}
\mathsf{FSelem} &\sqsubseteq \exists \mathsf{name}.\mathsf{String} \\
\mathsf{FSelem} &\equiv \mathsf{Directory} \sqcup \mathsf{File} \\
\mathsf{Directory} &\sqsubseteq \neg \mathsf{File} \\
\mathsf{Directory} &\sqsubseteq \forall \mathsf{child}.\mathsf{FSelem} \\
\mathsf{File} &\sqsubseteq \forall \mathsf{child}.\bot \\
\mathsf{Root} &\sqsubseteq \mathsf{Directory} \\
\mathsf{Root} &\sqsubseteq \forall \mathsf{child}^{-}.\bot
\end{aligned}$$

The axioms in $\mathcal{T}_{file}$ imply that in a model every object connected by a chain of role child to an instance of Root is an instance of FSelem. Formally, $\mathcal{T}_{file} \models \exists(\mathsf{child}^{-})^{*}.\mathsf{Root} \sqsubseteq \mathsf{FSelem}$. To verify that the implication holds, suppose that there exists a model in which an instance o of $\exists(\mathsf{child}^{-})^{*}.\mathsf{Root}$ is not an instance of FSelem. Then, reasoning by induction on the length of the chain from the instance of Root to o, one can derive a contradiction. Observe that induction is required, and hence such reasoning is not first-order.

2.3 Propositional Dynamic Logics

Propositional Dynamic Logics (PDLs) are modal logics specifically developed for reasoning about computer programs [FL79, KT90, HKT00]. Next we provide a brief overview of PDLs, and illustrate the correspondence between description logics and PDLs.

Syntactically, a PDL is constituted by expressions of two sorts: *programs* and *formulae*. Programs and formulae are built by starting from *atomic programs* and *propositional letters*, and applying suitable operators. We denote propositional letters with A, arbitrary formulae with ϕ, atomic programs with P, and arbitrary programs with r, all possibly with subscripts. We focus on *converse*-PDL [FL79] which, as it turns out, corresponds to $\mathcal{ALCI}_{reg}$. The abstract syntax of *converse*-PDL is as follows:

$$\begin{aligned}
\phi, \phi' &\longrightarrow \top \mid \bot \mid A \mid \phi \wedge \phi' \mid \phi \vee \phi' \mid \neg\phi \mid \langle r \rangle \phi \mid [r]\phi \\
r, r' &\longrightarrow P \mid r \cup r' \mid r; r' \mid r^{*} \mid \phi? \mid r^{-}
\end{aligned}$$

The basic Propositional Dynamic Logic PDL [FL79] is obtained from *converse*-PDL by dropping converse programs r^{-}.

The semantics of PDLs is based on the notion of (Kripke) structure, defined as a triple $\mathcal{M} = (\mathcal{S}, \{\mathcal{R}_P\}, \Pi)$, where $\mathcal{S}$ denotes a non-empty set of states, $\{\mathcal{R}_P\}$ is a family of binary relations over $\mathcal{S}$, each of which denotes the state transitions caused by an atomic program P, and Π is a mapping from $\mathcal{S}$ to propositional letters such that $\Pi(s)$ determines the letters that are true in state s. The basic semantical relation is "a formula ϕ holds at a state s of a structure $\mathcal{M}$", written $\mathcal{M}, s \models \phi$, and is defined by induction on the

formation of ϕ:

$$\begin{array}{lll}
\mathcal{M}, s \models A & \text{iff} & A \in \Pi(s) \\
\mathcal{M}, s \models \top & & \text{always} \\
\mathcal{M}, s \models \bot & & \text{never} \\
\mathcal{M}, s \models \phi \wedge \phi' & \text{iff} & \mathcal{M}, s \models \phi \text{ and } \mathcal{M}, s \models \phi' \\
\mathcal{M}, s \models \phi \vee \phi' & \text{iff} & \mathcal{M}, s \models \phi \text{ or } \mathcal{M}, s \models \phi' \\
\mathcal{M}, s \models \neg\phi & \text{iff} & \mathcal{M}, s \not\models \phi \\
\mathcal{M}, s \models \langle r \rangle \phi & \text{iff} & \text{there is } s' \text{ such that } (s, s') \in \mathcal{R}_r \text{ and } \mathcal{M}, s' \models \phi \\
\mathcal{M}, s \models [r]\phi & \text{iff} & \text{for all } s', (s, s') \in \mathcal{R}_r \text{ implies } \mathcal{M}, s' \models \phi
\end{array}$$

where the family $\{\mathcal{R}_P\}$ is systematically extended so as to include, for every program r, the corresponding relation $\mathcal{R}_r$ defined by induction on the formation of r:

$$\begin{array}{rcl}
\mathcal{R}_P & \subseteq & \mathcal{S} \times \mathcal{S} \\
\mathcal{R}_{r \cup r'} & = & \mathcal{R}_r \cup \mathcal{R}_{r'} \\
\mathcal{R}_{r;r'} & = & \mathcal{R}_r \circ \mathcal{R}_{r'} \\
\mathcal{R}_{r^*} & = & (\mathcal{R}_r)^* \\
\mathcal{R}_{\phi?} & = & \{(s, s) \in \mathcal{S} \times \mathcal{S} \mid \mathcal{M}, s \models \phi\} \\
\mathcal{R}_{r^-} & = & \{(s_1, s_2) \in \mathcal{S} \times \mathcal{S} \mid (s_2, s_1) \in \mathcal{R}_r\}.
\end{array}$$

If, for each atomic program P, the transition relation $\mathcal{R}_P$ is required to be a function that assigns to each state a unique successor state, then we are dealing with the *deterministic* variants of PDLs, namely DPDL and *converse*-DPDL [BAHP82, VW86].

A structure $\mathcal{M} = (\mathcal{S}, \{\mathcal{R}_P\}, \Pi)$ is called a *model* of a formula ϕ if there exists a state $s \in \mathcal{S}$ such that $\mathcal{M}, s \models \phi$. A formula ϕ is *satisfiable* if there exists a model of ϕ, otherwise the formula is *unsatisfiable*. A formula ϕ is *valid* in structure $\mathcal{M}$ if for all $s \in \mathcal{S}$, $\mathcal{M}, s \models \phi$. We call *axioms* formulae that are used to select the interpretations of interest. Formally, a structure $\mathcal{M}$ is a model of an axiom ϕ, if ϕ is valid in $\mathcal{M}$. A structure $\mathcal{M}$ is a model of a finite set of axioms Γ if $\mathcal{M}$ is a model of all axioms in Γ. An axiom is satisfiable if it has a model and a finite set of axioms is satisfiable if it has a model. We say that a finite set Γ of axioms *logically implies* a formula ϕ, written $\Gamma \models \phi$, if ϕ is valid in every model of Γ.

It is easy to see that satisfiability of a formula ϕ as well as satisfiability of a finite set of axioms Γ can be reformulated by means of logical implication, as $\emptyset \not\models \neg\phi$ and $\Gamma \not\models \bot$ respectively.

Interestingly, logical implication can, in turn, be reformulated in terms of satisfiability, by making use of the following theorem (cf. [KT90]).

Theorem 2.2 (Internalization of axioms) *Let Γ be a finite set of* converse-PDL *axioms, and ϕ a* converse-PDL *formula. Then $\Gamma \models \phi$ if and only if the formula*

$$\neg\phi \wedge [(P_1 \cup \cdots \cup P_m \cup P_1^- \cup \cdots \cup P_m^-)^*]\Gamma'$$

is unsatisfiable, where $P_1, \ldots, P_m$ are all atomic programs occurring in $\Gamma \cup \{\phi\}$ and Γ' is the conjunction of all axioms in Γ.

Such a result exploits the power of program constructs (union, reflexive-transitive closure) and the *connected model property* (i.e., if a formula has a model, it has a model which is connected) of PDLs in order to represent axioms. The connected model property is typical of modal logics and it is enjoyed by all PDLs. As a consequence, a result analogous to Theorem 2.2 holds for virtually all PDLs.

Reasoning in PDLs has been thoroughly studied from the computational point of view, and the results for the PDLs considered here are summarized in the following theorem [FL79, Pra79, BAHP82, VW86]:

Theorem 2.3 *Satisfiability in* PDL *is* ExpTime*-hard. Satisfiability in* PDL*, in* converse-PDL*, and in* converse-DPDL *can be decided in deterministic exponential time.*

2.4 The correspondence

The correspondence between description logics and PDLs was first published by in [Sch91], where it was shown that $\mathcal{ALCI}_{reg}$ can be considered a notational variant of *converse*-PDL. This observation allowed for exploiting the results on *converse*-PDL for instantly closing long standing issues regarding the decidability and complexity of both satisfiability and logical implication in $\mathcal{ALC}_{reg}$ and $\mathcal{ALCI}_{reg}$.[2] The paper was very influential for the research in expressive description logics in the following decade, since thanks to the correspondence between PDLs and description logics, first results but especially formal techniques and insights could be shared by the two communities. The correspondence between PDLs and description logics has been extensively used to study reasoning methods for expressive description logics. It has also lead to a number of interesting extensions of PDLs in terms of those constructs that are typical of description logics and have never been considered in PDLs. In particular, there is a tight relation between qualified number restrictions and graded modalities in modal logics [VdH92, VdHdR95, FBDC85, Fin72].

The correspondence is based on the similarity between the interpretation structures of the two logics: at the extensional level, individuals (members of $\Delta^{\mathcal{I}}$) in description logics correspond to states in PDLs, whereas links between two individuals correspond to state transitions. At the intensional level, concepts correspond to propositions, and roles correspond to programs. Formally, the correspondence is realized through a one-to-one and onto mapping τ from $\mathcal{ALCI}_{reg}$ concepts to *converse*-PDL formulae, and from $\mathcal{ALCI}_{reg}$ roles to *converse*-PDL programs. The mapping τ is defined inductively as follows:

$$
\begin{array}{rclcrcl}
\tau(A) & = & A & \qquad & \tau(P) & = & P \\
\tau(\neg C) & = & \neg\tau(C) & & \tau(R^-) & = & \tau(R)^- \\
\tau(C \sqcap C') & = & \tau(C) \wedge \tau(C') & & \tau(R \sqcup R') & = & \tau(R) \cup \tau(R') \\
\tau(C \sqcup C') & = & \tau(C) \vee \tau(C') & & \tau(R \circ R') & = & \tau(R); \tau(R') \\
\tau(\forall R.C) & = & [\tau(R)]\tau(C) & & \tau(R^*) & = & \tau(R)^* \\
\tau(\exists R.C) & = & \langle\tau(R)\rangle\tau(C) & & \tau(id(C)) & = & \tau(C)?
\end{array}
$$

Axioms in description logics' TBoxes correspond in the obvious way to axioms in PDLs. Moreover all forms of reasoning (satisfiability, logical implication, etc.) have their natural counterpart.

[2] In fact, the decidability of $\mathcal{ALC}_{reg}$ without the $id(C)$ construct was independently established in [Baa91].

One of the most important contributions of the correspondence is obtained by rephrasing Theorem 2.2 in terms of description logics. It says that every TBox can be "internalized" into a single concept, i.e., it is possible to build a concept that expresses all the axioms of the TBox. In doing so we rely on the ability to build a "universal" role, i.e., a role linking all individuals in a (connected) model. Indeed, a universal role can be expressed by using regular expressions over roles, and in particular the union of roles and the reflexive-transitive closure. The possibility of internalizing the TBox when dealing with expressive description logics tells us that for such description logics reasoning with TBoxes, i.e., logical implication, is no harder that reasoning with a single concept.

Theorem 2.4 *Concept satisfiability and logical implication in $\mathcal{ALC}_{reg}$ are* ExpTime*-hard. Concept satisfiability and logical implication in $\mathcal{ALC}_{reg}$ and $\mathcal{ALCI}_{reg}$ can be decided in deterministic exponential time.*

Observe that for description logics that do not allow for expressing a universal role, there is a sharp difference between reasoning techniques used in the presence of TBoxes, and techniques used to reason on concept expressions. The profound difference is reflected by the computational properties of the associated decision problems. For example, the logic $\mathcal{AL}$ admits simple structural algorithms for deciding reasoning tasks not involving axioms, and these algorithms are sound and complete and work in PTime. However, if general inclusion axioms are considered, then reasoning becomes ExpTime-complete, and the decision procedures that have been developed include suitable termination strategies [BDS93a]. Similarly, for the more expressive logic $\mathcal{ALC}$, reasoning tasks not involving a TBox are PSpace-complete [SSS91], while those that do involve it are ExpTime-complete.

2.5 Functional restrictions

We have seen that the logics $\mathcal{ALC}_{reg}$ and $\mathcal{ALCI}_{reg}$ correspond to standard PDL and *converse*-PDL respectively, which are both well studied. In fact the correspondence can be used to deal also with constructs that are typical of description logics, namely functional restrictions, by exploiting techniques developed for reasoning in PDLs. In particular, we can adopt automata-based techniques, which have been very successful in studying reasoning for expressive variants of PDL and characterizing their complexity.

Functional restrictions are the simplest form of number restrictions considered in description logics, and allow for specifying local functionality of roles, i.e., that instances of certain concepts have unique role-fillers for a given role. By adding functional restrictions on atomic roles and their inverse to $\mathcal{ALCI}_{reg}$, we obtain the description logic $\mathcal{ALCFI}_{reg}$. The PDL corresponding to $\mathcal{ALCFI}_{reg}$ is a PDL that extends *converse*-DPDL [VW86] with determinism of both atomic programs and their inverse, and such that determinism is no longer a global property, but one that can be imposed locally.

Formally, $\mathcal{ALCFI}_{reg}$ is obtained from $\mathcal{ALCI}_{reg}$ by adding *functional restrictions* of the form $\leqslant 1\, Q$, where Q is a *basic role*, i.e., either an atomic role or the inverse of an atomic role. Such a functional restriction is interpreted as follows:

$$(\leqslant 1\, Q)^{\mathcal{I}} \;=\; \{o \in \Delta^{\mathcal{I}} \mid |\{o' \in \Delta^{\mathcal{I}} \mid (o, o') \in Q^{\mathcal{I}}\}| \leq 1\}$$

We show that reasoning in $\mathcal{ALCFI}_{reg}$ is in ExpTime, and, since reasoning in $\mathcal{ALC}_{reg}$ is already ExpTime-hard, is in fact ExpTime-complete. Without loss of generality we concentrate on concept satisfiability. We exploit the fact that $\mathcal{ALCFI}_{reg}$ has the *tree model property*, which states that if a $\mathcal{ALCFI}_{reg}$ concept C is satisfiable then it is satisfied in an interpretation which has the structure of a (possibly infinite) tree with bounded branching degree. This allows us to make use of techniques based on automata on infinite trees. In particular, we can make use of *two-way alternating automata on infinite trees* (2ATAs) introduced in [Var98]. 2ATAs were used in [Var98] to derive a decision procedure for modal μ-calculus with backward modalities.

Theorem 2.5 *Concept satisfiability (and hence logical implication) in* $\mathcal{ALCFI}_{reg}$ *is* ExpTime*-complete.*

2.6 Qualified number restrictions

The most general form of number restrictions are the so-called *qualified number restrictions*, which allow for specifying arbitrary cardinality constraints on roles with role-fillers belonging to a certain concept. In particular we consider qualified number restrictions on basic roles, i.e., atomic roles and their inverse. By adding such constructs to $\mathcal{ALCI}_{reg}$ we obtain the description logic $\mathcal{ALCQI}_{reg}$. The PDL corresponding to $\mathcal{ALCQI}_{reg}$ is an extension of *converse*-PDL with "graded modalities" [FBDC85, VdHdR95, Tob99c] on atomic programs and their converse.

Formally, $\mathcal{ALCQI}_{reg}$ is obtained from $\mathcal{ALCI}_{reg}$ by adding *qualified number restrictions* of the form $\leqslant n\, QC$ and $\geqslant n\, QC$, where n is a nonnegative integer, Q is a basic role, and C is an $\mathcal{ALCQI}_{reg}$ concept. Such constructs are interpreted as follows:

$$\begin{aligned}(\leqslant n\, QC)^{\mathcal{I}} &= \{o \in \Delta^{\mathcal{I}} \mid |\{o' \in \Delta^{\mathcal{I}} \mid (o,o') \in Q^{\mathcal{I}} \wedge o' \in C^{\mathcal{I}}\}| \leq n\}\\ (\geqslant n\, QC)^{\mathcal{I}} &= \{o \in \Delta^{\mathcal{I}} \mid |\{o' \in \Delta^{\mathcal{I}} \mid (o,o') \in Q^{\mathcal{I}} \wedge o' \in C^{\mathcal{I}}\}| \geq n\}\end{aligned}$$

This can be shown by extending the automata theoretic techniques to deal also with qualified number restrictions. Also, a polynomial reduction from $\mathcal{ALCQI}_{reg}$ to $\mathcal{ALCFI}_{reg}$ was shown in [DGL95, DG95]. Such a reduction is based on the notion of *reification*, which plays a major role also in dealing with Boolean combinations of (atomic) roles [DGL95, DGL94b], as well as in extending expressive description logics with relation of arbitrary arity (see subsection 2.9).

Theorem 2.6 *Concept satisfiability (and hence logical implication) in* $\mathcal{ALCQI}_{reg}$ *is* ExpTime*-complete.*

We remind the reader that originally the ExpTime-completeness of reasoning in $\mathcal{ALCQI}_{reg}$ was shown under the standard assumption in description logics, that numbers in number restrictions are represented in unary. In [Tob01] techniques for dealing with qualified number restrictions with numbers coded in binary are presented, and are used to show that even under this assumption reasoning over $\mathcal{ALCQI}$ knowledge bases can be done in ExpTime. Such techniques extend to $\mathcal{ALCQI}_{reg}$ as shown in [CDGL02].

2.7 Objects

In this subsection, we review results involving knowledge on individuals expressed in terms of membership assertions. Given an alphabet $\mathcal{O}$ of symbols for individuals, a *(membership) assertion* has one of the following forms:

$$C(a) \qquad\qquad P(a_1, a_2)$$

where C is a concept, P is an atomic role, and a, a_1, a_2 belong to $\mathcal{O}$. An interpretation $\mathcal{I}$ is extended so as to assign to each $a \in \mathcal{O}$ an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ in such a way that the *unique name assumption* is satisfied, i.e., different elements are assigned to different symbols in $\mathcal{O}$. $\mathcal{I}$ *satisfies* $C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$, and $\mathcal{I}$ *satisfies* $P(a_1, a_2)$ if $(a_1^{\mathcal{I}}, a_2^{\mathcal{I}}) \in R^{\mathcal{I}}$. An *ABox* $\mathcal{A}$ is a finite set of membership assertions, and an interpretation $\mathcal{I}$ is called a *model of* $\mathcal{A}$ if $\mathcal{I}$ satisfies every assertion in $\mathcal{A}$.

A *knowledge base* is a pair $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, where $\mathcal{T}$ is a TBox, and $\mathcal{A}$ is an ABox. An interpretation $\mathcal{I}$ is called a *model of* $\mathcal{K}$ if it is a model of both $\mathcal{T}$ and $\mathcal{A}$. $\mathcal{K}$ is *satisfiable* if it has a model, and $\mathcal{K}$ *logically implies* an assertion β, denoted $\mathcal{K} \models \beta$, where β is either an inclusion or a membership assertion, if every model of $\mathcal{K}$ satisfies β. Logical implication can be reformulated in terms of unsatisfiability: e.g., $\mathcal{K} \models C(a)$ iff $\mathcal{K} \cup \{\neg C(a)\}$ is unsatisfiable; similarly $\mathcal{K} \models C_1 \sqsubseteq C_2$ iff $\mathcal{K} \cup \{(C_1 \sqcap \neg C_2)(a')\}$ is unsatisfiable, where a' does not occur in $\mathcal{K}$. Therefore, we only need a procedure for checking satisfiability of a knowledge base, whose complexity characterization is given below:

Theorem 2.7 *Knowledge base satisfiability (and hence every standard reasoning service) in* $\mathcal{ALCQI}_{reg}$ *is* ExpTime*-complete.*

The work in [DGL94a] and [DG95] extends $\mathcal{ALCQ}_{reg}$ and $\mathcal{ALCI}_{reg}$ by adding special atomic concepts A_a, called *nominals*, having exactly one single instance a, i.e., the individual they name. Nominals may occur in concepts exactly as atomic concepts, and hence they constitute one of the most flexible ways to express knowledge about single individuals.

By using nominals we can capture the "one-of" construct, having the form $\{a_1, \ldots, a_n\}$, denoting the concept made of exactly the enumerated individuals $a_1, \ldots, a_n$ [3]. We can also capture the "fills" construct, having the form $R : a$, denoting those individuals having the individual a as a role filler of R [4] (see [Sch94a] and references therein for further discussion on these constructs).

Let us denote with $\mathcal{ALCQO}_{reg}$ and $\mathcal{ALCIO}_{reg}$ the description logics resulting by adding nominals to $\mathcal{ALCQ}_{reg}$ and $\mathcal{ALCI}_{reg}$ respectively. [DGL94a] and [DG95] polynomially reduce satisfiability in $\mathcal{ALCQO}_{reg}$ and $\mathcal{ALCIO}_{reg}$ knowledge bases to satisfiability of $\mathcal{ALCQ}_{reg}$ and $\mathcal{ALCI}_{reg}$ concepts respectively, hence showing decidability and ExpTime-completeness of reasoning in these logics. ExpTime-completeness does not hold for $\mathcal{ALCQIO}_{reg}$, i.e., $\mathcal{ALCQI}_{reg}$ extended with nominals. Indeed, a result in

[3] Actually, nominals and the one-of construct are essentially equivalent, since a name A_a is equivalent to $\{a\}$ and $\{a_1, \ldots, a_n\}$ is equivalent to $A_{a_1} \sqcup \cdots \sqcup A_{a_n}$.

[4] The "fills" construct $R : a$ is captured by $\exists R.A_a$.

[Tob99a, Tob99b] shows that reasoning in such a logic is NExpTime-hard. Its decidability still remains an open problem.

The notion of nominal introduced above has a correspondent in modal logic [Pri67, Bul70, BS93, GG93, Bla93]. Nominals have also been studied within the setting of PDLs [PT85, GP88, PT91]. The results for $\mathcal{ALCQO}_{reg}$ and $\mathcal{ALCIO}_{reg}$ are immediately applicable also in the setting of PDLs. In particular, the PDL corresponding to $\mathcal{ALCQO}_{reg}$ is standard PDL augmented with nominals and graded modalities (qualified number restrictions). It is an extension of *deterministic combinatory PDL*, DcPDL, which is essentially DPDL augmented with nominals. The decidability of DcPDL is established in [PT85], who also prove that satisfiability can be checked in nondeterministic double exponential time. This is tightened by the result above on ExpTime-completeness of $\mathcal{ALCQO}_{reg}$, which says that DcPDL is in fact ExpTime-complete, thus closing the previous gap between the upper bound and the lower bound. The PDL corresponding to $\mathcal{ALCIO}_{reg}$ is *converse*-PDL augmented with nominals, which is also called *converse combinatory PDL*, CcPDL [PT91]. Such logic was not known to be decidable [PT91]. Hence the results mentioned above allow us to establish the decidability of CcPDL and to precisely characterize the computational complexity of satisfiability (and hence of logical implication) as ExpTime-complete.

2.8 Fixpoint constructs

Decidable description logics equipped with explicit fixpoint constructs have been devised in order to model inductive and coinductive data structures such as lists, streams, trees, etc. [DGL94c, Sch94b, DGL97, CDGL99b]. Such logics correspond to extensions of the *propositional μ-calculus* [Koz83, SE89, Var98], a variant of PDL with explicit fixpoints that is used to express temporal properties of reactive and concurrent processes [Sti96, Eme96]. Such logics can also be viewed as a well-behaved fragment of first-order logic with fixpoints [Par70, Par76, AHV95].

Here, we concentrate on the description logic $\mu\mathcal{ALCQI}$ studied in [CDGL99b]. Such a description logic is derived from $\mathcal{ALCQI}$ by adding *least and greatest fixpoint constructs*. The availability of explicit fixpoint constructs allows for expressing *inductive* and *coinductive* concepts in a natural way.

Example 2.8 Consider the concept Tree, representing trees, inductively defined as follows:

1. An individual that is an EmptyTree is a Tree.
2. If an individual is a Node, has at most one parent, has some children, and all children are Trees, then such an individual is a Tree.

In other words, Tree is the concept with the smallest extension among those satisfying the assertions 1 and 2. Such a concept is naturally expressed in $\mu\mathcal{ALCQI}$ by making use of the least fixpoint construct $\mu X.C$:

$$\mathsf{Tree} \equiv \mu X.(\mathsf{EmptyTree} \sqcup (\mathsf{Node} \sqcap \leqslant 1\, \mathsf{child}^{-} \sqcap \exists \mathsf{child}.\top \sqcap \forall \mathsf{child}.X))$$

Example 2.9 Consider the well-known linear data structure, called stream. Streams are similar to lists except that, while lists can be considered as finite sequences of nodes, streams are infinite sequences of nodes. Such a data structure is captured by the concept Stream, coinductively defined as follows:

1. An individual that is a Stream, is a Node and has a single successor which is a Stream.

In other words, Stream is the concept with the largest extension among those satisfying condition 1. Such a concept is naturally expressed in $\mu\mathcal{ALCQI}$ by making use of the greatest fixpoint construct $\nu X.C$:

$$\mathsf{Stream} \equiv \nu X.(\mathsf{Node} \sqcap \leqslant 1\,\mathsf{succ} \sqcap \exists \mathsf{succ}.X)$$

Let us now introduce $\mu\mathcal{ALCQI}$ formally. We make use of the standard first-order notions of *scope*, *bound* and *free occurrences* of variables, *closed formulae*, etc., treating μ and ν as quantifiers.

The primitive symbols in $\mu\mathcal{ALCQI}$ are *atomic concepts*, *(concept) variables*, and *atomic roles*. Concepts and roles are formed according to the following syntax

$$\begin{array}{lcl} C & \longrightarrow & A \mid \neg C \mid C_1 \sqcap C_2 \mid \geqslant n\, R.C \mid \mu X.C \mid X \\ R & \longrightarrow & P \mid P^- \end{array}$$

where A denotes an atomic concept, P an atomic role, C an arbitrary $\mu\mathcal{ALCQI}$ concept, R an arbitrary $\mu\mathcal{ALCQI}$ role (i.e., either an atomic role or the inverse of an atomic role), n a natural number, and X a variable.

The concept C in μXC must be *syntactically monotone*, that is, every free occurrence of the variable X in C must be in the scope of an even number of negations [Koz83]. This restriction guarantees that the concept C denotes a monotonic operator and hence both the least and the greatest fixpoints exist and are unique (see later).

In addition to the usual abbreviations used in $\mathcal{ALCQI}$, we introduce the abbreviation $\nu X.C$ for $\neg\mu X.\neg C[X/\neg X]$, where $C[X/\neg X]$ is the concept obtained by substituting all free occurrences of X with $\neg X$.

The presence of free variables does not allow us to extend the interpretation function $\cdot^{\mathcal{I}}$ directly to every concept of the logic. For this reason we introduce valuations. A *valuation* ρ on an interpretation $\mathcal{I}$ is a mapping from variables to subsets of $\Delta^{\mathcal{I}}$. Given a valuation ρ, we denote by $\rho[X/\mathcal{E}]$ the valuation identical to ρ except for the fact that $\rho[X/\mathcal{E}](X) = \mathcal{E}$.

Let $\mathcal{I}$ be an interpretation and ρ a valuation on $\mathcal{I}$. We assign meaning to concepts of the logic by associating to $\mathcal{I}$ and ρ an *extension function* $\cdot^{\mathcal{I}}_{\rho}$, mapping concepts to subsets of $\Delta^{\mathcal{I}}$, as follows:

$$
\begin{aligned}
X^{\mathcal{I}}_{\rho} &= \rho(X) \subseteq \Delta^{\mathcal{I}} \\
A^{\mathcal{I}}_{\rho} &= A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}}_{\rho} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}_{\rho} \\
(C_1 \sqcap C_2)^{\mathcal{I}}_{\rho} &= (C_1)^{\mathcal{I}}_{\rho} \cap (C_2)^{\mathcal{I}}_{\rho} \\
\geqslant n\, R.C^{\mathcal{I}}_{\rho} &= \{s \in \Delta^{\mathcal{I}} \mid |\{s' \mid (s,s') \in R^{\mathcal{I}} \text{ and } s' \in C^{\mathcal{I}}_{\rho}\}| \geq n\} \\
(\mu X.C)^{\mathcal{I}}_{\rho} &= \bigcap \{\mathcal{E} \subseteq \Delta^{\mathcal{I}} \mid\ C^{\mathcal{I}}_{\rho[X/\mathcal{E}]} \subseteq \mathcal{E}\ \}
\end{aligned}
$$

Observe that $C^{\mathcal{I}}_{\rho[X/\mathcal{E}]}$ can be seen as an operator from subsets $\mathcal{E}$ of $\Delta^{\mathcal{I}}$ to subsets of $\Delta^{\mathcal{I}}$, and that, by the syntactic restriction enforced on variables, such an operator is guaranteed to be monotonic w.r.t. set inclusion. $\mu X.C$ denotes the *least fixpoint* of the operator. Observe also that the semantics assigned to $\nu X.C$ is

$$(\nu X.C)^{\mathcal{I}}_{\rho} = \bigcup \{\mathcal{E} \subseteq \Delta^{\mathcal{I}} \mid\ \mathcal{E} \subseteq\ C^{\mathcal{I}}_{\rho[X/\mathcal{E}]}\ \}$$

Hence $\nu X.C$ denotes the *greatest fixpoint* of the operator.

In fact, we are interested in closed concepts, whose extension is independent of the valuation. For closed concepts we do not need to consider the valuation explicitly, and hence the notion of concept satisfiability, logical implication, etc. extend straightforwardly.

Exploiting a recent result on ExpTime decidability of modal μ-calculus with converse [Var98], and exploiting a reduction technique for qualified number restrictions similar to the one mentioned in subsection 2.6, [CDGL99b] has shown that the same complexity bound holds also for reasoning in $\mu\mathcal{ALCQI}$.

Theorem 2.10 *Concept satisfiability (and hence logical implication) in $\mu\mathcal{ALCQI}$ is* ExpTime*-complete.*

For certain applications, variants of $\mu\mathcal{ALCQI}$ that allow for *mutual fixpoints*, denoting least and greatest solutions of *mutually* recursive equations, are of interest [Sch94b, CDGL98b, CDGL99a]. Mutual fixpoints can be re-expressed by suitably nesting the kind of fixpoints considered here (see, for example, [dB80, Sch94b]). It is interesting to notice that, although the resulting concept may be exponentially large in the size of the original concept with mutual fixpoints, the number of (distinct) subconcepts of the resulting concept is polynomially bounded by the size of the original one. By virtue of this observation, and using the reasoning procedure in [CDGL99b], we can strengthen the above result.

Theorem 2.11 *Checking satisfiability of a closed $\mu\mathcal{ALCQI}$ concept C can be done in deterministic exponential time w.r.t. the number of (distinct) subconcepts of C.*

Although $\mu\mathcal{ALCQI}$ does not have the rich variety of role constructs of $\mathcal{ALCQI}_{reg}$, it is actually an extension of $\mathcal{ALCQI}_{reg}$, since any $\mathcal{ALCQI}_{reg}$ concept can be expressed in

$\mu\mathcal{ALCQI}$ using the fixpoint constructs in a suitable way. To express concepts involving complex role expressions, it suffices to resort to the following equivalences:

$$\begin{aligned} \exists(R_1 \circ R_2).C &= \exists R_1.\exists R_2.C \\ \exists(R_1 \sqcup R_2).C &= \exists R_1.C \sqcup \exists R_2.C \\ \exists R^*.C &= \mu X.(C \sqcup \exists R.X) \\ \exists id(D).C &= C \sqcap D. \end{aligned}$$

Note that, according to such equivalences, we have also that

$$\forall R^*.C = \nu X.(C \sqcap \forall R.X)$$

[CDGL95] advocate a further construct corresponding to an implicit form of fixpoint, the so called *well-founded* concept construct $wf(R)$. Such construct is used to impose well-foundedness of chains of roles, and thus allows one to correctly capture inductive structures. Using explicit fixpoints, $wf(R)$ is expressed as $\mu X.(\forall R.X)$.

We remark that, in order to gain the ability of expressing inductively and coinductively defined concepts, it has been proposed to adopt ad hoc semantics for interpreting knowledge bases, specifically the *least fixpoint semantics* for expressing inductive concepts and the *greatest fixpoint semantics* for expressing coinductive ones (see [BCM+03, Neb91, Baa90, Baa91, DMO92, Küs98, BDNS98]). Logics equipped with fixpoint constructs allow for mixing statements interpreted according to the least and greatest fixpoint semantics in the same knowledge base [Sch94b, DGL97], and thus can be viewed as a generalization of these approaches.

Recently, using techniques based on alternating two-way automata, it has been shown that the propositional μ-calculus with converse programs remains ExpTime-decidable when extended with nominals [SV01]. Such a logic corresponds to a description logic which could be called $\mu\mathcal{ALCIO}$.

2.9 Relations of arbitrary arity

A limitation of traditional description logics is that only binary relationships between instances of concepts can be represented, while in some real world situations it is required to model relationships among more than two objects. Such relationships can be captured by making use of relations of arbitrary arity instead of (binary) roles. Various extensions of description logics with relations of arbitrary arity have been proposed [Sch89, CL93, DGL94b, CDGL97, CDGL98a, LST99].

We concentrate on the description logic $\mathcal{DLR}$ [CDGL97, CDGL98a], which represents a natural generalization of traditional description logics towards n-ary relations. The basic elements of $\mathcal{DLR}$ are *atomic relations* and *atomic concepts*, denoted by $\mathbf{P}$ and A respectively. Arbitrary *relations*, of given *arity* between 2 and n_{max}, and arbitrary *concepts* are formed according to the following syntax

$$\begin{aligned} \mathbf{R} &\longrightarrow \top_n \mid \mathbf{P} \mid (\$i/n : C) \mid \neg\mathbf{R} \mid \mathbf{R}_1 \sqcap \mathbf{R}_2 \\ C &\longrightarrow \top_1 \mid A \mid \neg C \mid C_1 \sqcap C_2 \mid \exists[\$i]\mathbf{R} \mid \leqslant k\,[\$i]\mathbf{R} \end{aligned}$$

where i and j denote components of relations, i.e., integers between 1 and n_{max}, n denotes the arity of a relation, i.e., an integer between 2 and n_{max}, and k denotes a nonnegative

integer. Concepts and relations must be *well-typed*, which means that only relations of the same arity n can be combined to form expressions of type $\mathbf{R}_1 \sqcap \mathbf{R}_2$ (which inherit the arity n), and $i \leq n$ whenever i denotes a component of a relation of arity n.

The semantics of $\mathcal{DLR}$ is specified through the usual notion of *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where the *interpretation function* $\cdot^{\mathcal{I}}$ assigns to each concept C a subset $C^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, and to each relation $\mathbf{R}$ of arity n a subset $\mathbf{R}^{\mathcal{I}}$ of $(\Delta^{\mathcal{I}})^n$, such that the following conditions are satisfied

$$\begin{array}{rcl}
\top_n^{\mathcal{I}} & \subseteq & (\Delta^{\mathcal{I}})^n \\
\mathbf{P}^{\mathcal{I}} & \subseteq & \top_n^{\mathcal{I}} \\
(\neg \mathbf{R})^{\mathcal{I}} & = & \top_n^{\mathcal{I}} \setminus \mathbf{R}^{\mathcal{I}} \\
(\mathbf{R}_1 \sqcap \mathbf{R}_2)^{\mathcal{I}} & = & \mathbf{R}_1^{\mathcal{I}} \cap \mathbf{R}_2^{\mathcal{I}} \\
(\$i/n\colon C)^{\mathcal{I}} & = & \{(d_1, \ldots, d_n) \in \top_n^{\mathcal{I}} \mid d_i \in C^{\mathcal{I}}\} \\
\\
\top_1^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \\
A^{\mathcal{I}} & \subseteq & \Delta^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C_1 \sqcap C_2)^{\mathcal{I}} & = & C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} \\
(\exists[\$i]\mathbf{R})^{\mathcal{I}} & = & \{d \in \Delta^{\mathcal{I}} \mid \exists (d_1, \ldots, d_n) \in \mathbf{R}^{\mathcal{I}}.\, d_i = d\} \\
(\leqslant k\,[\$i]\mathbf{R})^{\mathcal{I}} & = & \{d \in \Delta^{\mathcal{I}} \mid |\{(d_1, \ldots, d_n) \in \mathbf{R}_1^{\mathcal{I}} \mid d_i = d\}| \leq k\}
\end{array}$$

where $\mathbf{P}$, $\mathbf{R}$, $\mathbf{R}_1$, and $\mathbf{R}_2$ have arity n. Observe that $\top_1$ denotes the interpretation domain, while $\top_n$, for $n > 1$, does *not* denote the n-cartesian product of the domain, but only a subset of it, that covers all relations of arity n that are introduced. As a consequence, the "$\neg$" construct on relations expresses *difference of relations* rather than complement.

The construct $(\$i/n\colon C)$ denotes all tuples in $\top_n$ that have an instance of concept C as their i-th component, and therefore represents a kind of selection. Existential quantification and number restrictions on relations are a natural generalization of the corresponding constructs using roles. This can be seen by observing that, while for roles the "direction of traversal" is implicit, for a relation one needs to explicitly say which component is used to "enter" a tuple and which component is used to "exit" it.

$\mathcal{DLR}$ is in fact a proper generalization of $\mathcal{ALCQI}$. The traditional description logic constructs can be reexpressed in $\mathcal{DLR}$ as follows:

$$\begin{array}{lcl}
\exists P.C & \text{as} & \exists[\$1](P \sqcap (\$2/2\colon C)) \\
\exists P^-.C & \text{as} & \exists[\$2](P \sqcap (\$1/2\colon C)) \\
\forall P.C & \text{as} & \neg\exists[\$1](P \sqcap (\$2/2\colon \neg C)) \\
\forall P^-.C & \text{as} & \neg\exists[\$2](P \sqcap (\$1/2\colon \neg C)) \\
\leqslant k\, P.C & \text{as} & \leqslant k\,[\$1](P \sqcap (\$2/2\colon C)) \\
\leqslant k\, P^-.C & \text{as} & \leqslant k\,[\$2](P \sqcap (\$1/2\colon C))
\end{array}$$

Observe that the constructs using direct and inverse roles are represented in $\mathcal{DLR}$ by using binary relations and explicitly specifying the direction of traversal.

A TBox in $\mathcal{DLR}$ is a finite set of inclusion axioms on both concepts and relations of the form

$$C \sqsubseteq C' \qquad\qquad \mathbf{R} \sqsubseteq \mathbf{R}'$$

where $\mathbf{R}$ and $\mathbf{R}'$ are two relations of the same arity. The notions of an interpretation *satisfying* an assertion, and of *model* of a TBox are defined as usual.

The basic technique used in $\mathcal{DLR}$ to reason on relations is *reification*, which allows one to reduce logical implication in $\mathcal{DLR}$ to logical implication in $\mathcal{ALCQI}$. Reification for n-ary relations, which is similar to reification of roles (cf. 2.6): A relation of arity n is reified by means of a new concept and n functional roles $f_1, \ldots, f_n$. Let the $\mathcal{ALCQI}$ TBox $\mathcal{T}'$ be the *reified counterpart* of a $\mathcal{DLR}$ TBox $\mathcal{T}$. A tuple of a relation R in a model of $\mathcal{T}$ is represented in a model of $\mathcal{T}'$ by an instance of the concept corresponding to R, which is linked through $f_1, \ldots, f_n$ respectively to n individuals representing the components of the tuple. In this case reification is further used to encode Boolean constructs on relations into the corresponding constructs on the concepts representing relations.

Performing the reification of relations requires some attention, since the semantics of a relation rules out that there may be two identical tuples in its extension, i.e., two tuples constituted by the same components in the same positions. In the reified counterpart, on the other hand, one cannot explicitly rule out (e.g., by using specific axioms) the existence of two individuals o_1 and o_2 "representing" the same tuple, i.e., that are connected through $f_1, \ldots, f_n$ to exactly the same individuals denoting the components of the tuple. A model of the reified counterpart $\mathcal{T}'$ of $\mathcal{T}$ in which this situation occurs may not correspond directly to a model of $\mathcal{T}$, since by collapsing the two equivalent individuals into a tuple, axioms may be violated (e.g., cardinality constraints). However, also in this case a result holds, ensuring that from any model of $\mathcal{T}'$ one can construct a new one in which no two individuals represent the same tuple. Therefore one does not need to take this constraint explicitly into account when reasoning on the reified counterpart of a knowledge base with relations. Since reification is polynomial, from ExpTime decidability of logical implication in $\mathcal{ALCQI}$ (and ExpTime-hardness of logical implication in $\mathcal{ALC}$) we get the following characterization of the computational complexity of reasoning in $\mathcal{DLR}$ [CDGL97]

Theorem 2.12 *Logical implication in $\mathcal{DLR}$ is* ExpTime*-complete.*

$\mathcal{DLR}$ can be extended to include regular expressions built over projections of relations on two of their components, thus obtaining $\mathcal{DLR}_{reg}$. Such a logic, which represents a generalization of $\mathcal{ALCQI}_{reg}$, allows for the internalization of a TBox. ExpTime decidability (and hence completeness) of $\mathcal{DLR}_{reg}$ can again be shown by exploiting reification of relations and reducing logical implication to concept satisfiability in $\mathcal{ALCQI}_{reg}$ [CDGL98a]. Recently, $\mathcal{DLR}_{reg}$ has been extended to $\mathcal{DLR}_{\mu}$, which includes explicit fixpoint constructs on concepts, as those introduced in Section 2.8. The ExpTime-decidability result extends to $\mathcal{DLR}_{\mu}$ as well [CDGL99b].

Recently it has been observed that guarded fragments of first order logic [AvBN96, Grä99], which include n-ary relations, share with description logics the "locality" of quantification. This makes them of interest as extensions of description logics with n-ary relations [Grä98, LST99]. Such description logics are incomparable in expressive power with $\mathcal{DLR}$ and its extensions: On the one hand the description logics corresponding to guarded fragments allow one to refer, by the use of explicit variables, to components of relations in a more flexible way than what is possible in $\mathcal{DLR}$. On the other hand such description logics lack number restrictions, and extending them with number restrictions leads to undecidability of reasoning. Also, reasoning in the guarded fragments is in general

NExpTime-hard [Grä98, Grä99] and thus more difficult than in $\mathcal{DLR}$ and its extensions, although PSpace-complete fragments have been identified [LST99].

2.10 Boolean constructs on roles and role inclusion axioms

Observe also that $\mathcal{DLR}$ (and $\mathcal{DLR}_{reg}$) allows for Boolean constructs on relations (with negation interpreted as difference) as well as relation inclusion axioms $\mathbf{R} \sqsubseteq \mathbf{R}'$. In fact, $\mathcal{DLR}$ (resp. $\mathcal{DLR}_{reg}$) can be viewed as a generalization of $\mathcal{ALCQI}$ (resp. $\mathcal{ALCQI}_{reg}$) extended with Boolean constructs on atomic and inverse atomic roles. Such extensions of $\mathcal{ALCQI}$ were first studied in [DGL94b, DG95], where logical implication was shown to be ExpTime-complete by a reduction to $\mathcal{ALCQI}$ (resp. $\mathcal{ALCQI}_{reg}$). The logics above do not allow for combining atomic roles with inverse roles in Boolean combinations and role inclusion axioms. [Tob01] shows that, for $\mathcal{ALCQI}$ extended with arbitrary Boolean combinations of atomic and inverse atomic roles, logical implication remains in ExpTime. Decidability of PDL with Intersection and Converse is shown in [Lut05]. Note that, in all logics above, negation on roles is interpreted as difference. For results on the impact of full negation on roles see [LS01, Tob01].

[HST00a] investigate reasoning in $\mathcal{SHIQ}$, which is $\mathcal{ALCQI}$ extended with roles that are transitive and with role inclusion axioms on arbitrary roles (direct, inverse, and transitive). $\mathcal{SHIQ}$ does not include reflexive-transitive closure. However, transitive roles and role inclusions allow for expressing a universal role (in a connected model), and hence allow for internalizing TBoxes. Satisfiability and logical implication in $\mathcal{SHIQ}$ are ExpTime-complete [Tob01]. The importance of $\mathcal{SHIQ}$ lies in the fact that it is the logic implemented by the current state-of-the-art description logic-based systems (cf. next section).

2.11 Structured objects

An alternative way to overcome the limitations that result from the restriction to binary relationships between concepts, is to consider the interpretation domain as being constituted by objects with a complex structure, and extend the description logics with constructs that allow one to specify such structure [DGL95]. This approach is in the spirit of object-oriented data models used in databases [LR89, BK89, Hul88], and has the advantage, with respect to introducing relationships, that all aspects of the domain to be modeled can be represented in a uniform way, as concepts whose instances have certain structures. In particular, objects can either be unstructured or have the structure of a *set* or of a *tuple*. For objects having the structure of a set a particular role allows one to refer to the members of the set, and similarly each component of a tuple can be referred to by means of the (implicitly functional) role that labels it.

In general, reasoning over structured objects can have a very high computational complexity [KV93]. However, reasoning over a significant fragment of structuring properties can be polynomial reduced to reasoning in traditional description logics, by exploiting again reification to deal with tuples and sets. Thus, for such a fragment, reasoning can be done in ExpTime [DGL95]. An important aspect in exploiting description logics for reasoning over structured objects, is being able to limit the depth of the structure of an

object to avoid infinite nesting of tuples or sets. This requires the use of a well-founded construct, which is a restricted form of fixpoint (see Section 2.8).

2.12 Finite model reasoning

For expressive description logics, in particular for those containing inverse roles and functionality, a TBox may admit only models with an infinite domain [CKV90, CLN94]. Similarly, there may be TBoxes in which a certain concept can be satisfied only in an infinite model. This is illustrated in the following example in [Cal96c].

Example 2.13 Consider the TBox

$$\begin{aligned} \mathsf{FirstGuard} &\sqsubseteq \mathsf{Guard} \sqcap \forall \mathsf{shields}^{-}.\bot \\ \mathsf{Guard} &\sqsubseteq \exists \mathsf{shields} \sqcap \forall \mathsf{shields}.\mathsf{Guard} \sqcap \leqslant 1\, \mathsf{shields}^{-} \end{aligned}$$

In a model of this TBox, an instance of FirstGuard can have no shields-predecessor, while each instance of Guard can have at most one. Therefore, the existence of an instance of FirstGuard implies the existence of an infinite sequence of instances of Guard, each one connected through the role shields to the following one. This means that FirstGuard can be satisfied in an interpretation with a domain of arbitrary cardinality, but not in interpretations with a finite domain.

Note that the TBox above is expressed in a very simple description logic, in particular $\mathcal{AL}$ extended with inverse roles and functionality.

A logic is said to have the *finite model property* if every satisfiable formula of the logic admits a *finite model*, i.e., a model with a finite domain. The example above shows that virtually all description logics including functionality, inverse roles, and TBox axioms (or having the ability to internalize them) lack the finite model property. The example shows also that to lose the finite model property, functionality in only one direction is sufficient. In fact, it is well known that *converse*-DPDL, which corresponds to a fragment of $\mathcal{ALCFI}_{reg}$, lacks the finite model property [KT90, VW86].

For all logics that lack the finite model property, reasoning with respect to unrestricted and finite models are fundamentally different tasks, and this needs to be taken explicitly into account when devising reasoning procedures. Restricting reasoning to finite domains is not common in knowledge representation. However, it is typically of interest in databases, where one assumes that the data available are always finite [CLN94, CLN99].

When reasoning w.r.t. finite models, some properties that are essential for the techniques developed for unrestricted model reasoning in expressive description logics fail. In particular, all reductions exploiting the tree model property (or similar properties that are based on "unraveling" structures) [Var97] cannot be applied since this property does not hold when only finite models are considered. An intuitive justification can be given by observing that, whenever a (finite) model contains a cycle, the unraveling of such a model into a tree generates an infinite structure. Therefore alternative techniques have been developed.

Next, we look at the decidability and computational complexity of finite model reasoning over TBoxes expressed in various sublanguages of $\mathcal{ALCQI}$. Specifically, by using

techniques based on reductions to linear programming problems, it has been shown that finite concept satisfiability w.r.t. to $\mathcal{ALUNI}$ TBoxes[5] constituted by inclusion axioms only is ExpTime-complete [CLN94], and that finite model reasoning in arbitrary $\mathcal{ALCQI}$ TBoxes can be done in deterministic double exponential time [Cal96a]. In [LST05] it has been shown that finite model reasoning in $\mathcal{ALCQI}$ is ExpTime-complete even with binary coding of numbers.

For more expressive description logics, and in particular for all those description logics containing the construct for reflexive-transitive closure of roles, the decidability of finite model reasoning is still an open problem. Decidability of finite model reasoning for $\mathcal{C}^2$, i.e., first order logic with two variables and counting quantifiers was shown in [GOR97]. $\mathcal{C}^2$ is a logic that is strictly more expressive than $\mathcal{ALCQI}$ TBoxes, since it allows, for example, to impose cardinality restrictions on concepts [BBH96] or to use the full negation of a role. In [PH05b] it has been shown that finite and infinite model reasoning of $\mathcal{C}^2$ is NExpTime-complete even with binary coding of numbers.

Techniques for finite model reasoning have also been studied in databases. In the relational model, the interaction between inclusion dependencies and functional dependencies causes the loss of the finite model property, and finite implication of dependencies under various assumptions has been investigated in [CKV90]. A method for finite model reasoning has been presented in [CL94b, CL94a] in the context of a semantic and an object-oriented database model, respectively. The reasoning procedure, which represents a direct generalization of the one discussed above to relations of arbitrary arity, does not exploit reification to handle relations (see Section 2.9) but encodes directly the constraints on them into a system of linear inequalities.

2.13 Undecidability results

Several additional description logic constructs besides those discussed in the previous subsections have been proposed in the literature. Here, we present the most important of these extensions, discussing how they influence decidability, and what modifications to the reasoning procedures are needed to take them into account. In particular, we discuss Boolean constructs on roles, variants of role-value-maps or role agreements, and number restrictions on complex roles. Most of these constructs lead to undecidability of reasoning, if used in an unrestricted way. Roughly speaking, this is mainly due to the fact that the tree model property is lost [Var97].

2.13.1 Boolean constructs on complex roles

In those description logics that include regular expressions over roles, such as $\mathcal{ALCQI}_{reg}$, since regular languages are closed under intersection and complementation, the intersection of roles and the complement of a role are already expressible, if we consider them applied to the set of role expressions. Here we consider the more common approach in PDLs, namely to regard Boolean operators as applied to the binary relations denoted by complex roles. The logics thus obtained are more expressive than traditional PDL [Har84]

[5] $\mathcal{ALUNI}$ is the description logic obtained by extending $\mathcal{ALUN}$ (cf. [BCM+03] with inverse roles.

and reasoning is usually harder. We notice that the semantics immediately implies that intersection of roles can be expressed by means of union and complementation.

Satisfiability in PDL augmented with intersection of arbitrary programs is decidable in deterministic double exponential time [Dan84], and thus is satisfiability in $\mathcal{ALC}_{reg}$ augmented with intersection of complex roles, even though these logics have neither the tree nor the finite model property. On the other hand, satisfiability in PDL augmented with complementation of programs is undecidable [Har84], and so is reasoning in $\mathcal{ALC}_{reg}$ augmented with complementation of complex roles. Also, DPDL augmented with intersection of complex roles is highly undecidable [Har85, Har86], and since global functionality of roles (which corresponds to determinism of programs) can be expressed by means of local functionality, the undecidability carries over to $\mathcal{ALCF}_{reg}$ augmented with intersection of roles.

These proofs of undecidability make use of a general technique based on the reduction from the unbounded *tiling* (or *domino*) *problem* [Ber66, Rob71], which is the problem of checking whether a quadrant of the integer plane can be tiled using a finite set of tile types—i.e., square tiles with a color on each side—in such a way that adjacent tiles have the same color on the sides that touch[6]. We sketch the idea of the proof using the terminology of description logics, instead of that of PDLs. The reduction uses two roles right and up which are globally functional (i.e., $\leqslant 1\,\mathsf{right}$, $\leqslant 1\,\mathsf{up}$) and denote pairs of tiles that are adjacent in the x and y directions, respectively. By means of intersection of roles, right and up are constrained to effectively define a two-dimensional grid. This is achieved by imposing for each point of the grid (i.e., reachable through right and up) that by following $\mathsf{right} \circ \mathsf{up}$ one reaches a point reached also by following $\mathsf{up} \circ \mathsf{right}$:

$$\forall(\mathsf{right} \sqcup \mathsf{up})^*.\exists((\mathsf{right} \circ \mathsf{up}) \sqcap (\mathsf{up} \circ \mathsf{right}))$$

To enforce this condition, the use of intersection of compositions of atomic roles is essential. Reflexive-transitive closure (i.e., $\forall(\mathsf{right} \sqcup \mathsf{up})^*.C$) is then also exploited to impose the required constraints on all tiles of the grid. Observe that, in the above reduction, one can use TBox axioms instead of reflexive-transitive closure to enforce the necessary conditions in every point of the grid.

The question arises if decidability can be preserved if one restricts Boolean operations to basic roles, i.e., atomic roles and their inverse. This is indeed the case if complementation of basic roles is used only to express difference of roles, as demonstrated by the ExpTime decidability of $\mathcal{DLR}$ and its extensions, in which intersection and difference of relations are allowed (see Section 2.9).

2.13.2 Role-value-maps

Another construct, which stems from frame-systems, and which provides additional useful means to specify structural properties of concepts, is the so called *role-value-map* [BS85], which comes in two forms: An *equality role-value-map*, denoted $R_1 = R_2$, represents the individuals o such that the set of individuals that are connected to o via role R_1 equals the set of individuals connected to o via role R_2. The second form of role-value-map is

[6] In fact the reduction is from the Π^1_1-complete—and thus highly undecidable—recurring tiling problem [Har86], where one additionally requires that a certain tile occurs infinitely often on the x-axis.

containment role-value-map, denoted $R_1 \subseteq R_2$, whose semantics is defined analogously, using set inclusion instead of set equality. Using these constructs, one can denote, for example, by means of $\mathsf{owns} \circ \mathsf{made\_in} \subseteq \mathsf{lives\_in}$ the set of all persons that own only products manufactured in the country they live in.

When role-value-maps are added, the logic loses the tree model property, and this construct leads immediately to undecidability of reasoning when applied to *role chains* (i.e., compositions of atomic roles). For $\mathcal{ALC}_{reg}$, this can be shown by a reduction from the tiling problem in a similar way as to what is done in [Har85] for DPDL with intersection of roles. In this case, the concept $\mathsf{right} \circ \mathsf{up} = \mathsf{up} \circ \mathsf{right}$ involving role-value-map can be used instead of role intersection to define the constraints on the grid. The proof is slightly more involved than that for DPDL, since one needs to take into account that the roles right and up are not functional (while in DPDL all programs/roles are functional). However, undecidability holds already for concept subsumption (with respect to an empty TBox) in $\mathcal{AL}$ (in fact $\mathcal{FL}^-$) augmented with role-value-maps, where the involved roles are compositions of atomic roles [SS89].

As for role intersection, in order to show undecidability, it is necessary to apply role-value-maps to compositions of roles. Indeed, if the application of role-value-maps is restricted to Boolean combinations of basic roles, it can be added to $\mathcal{ALCQI}_{reg}$ without influencing decidability and worst case complexity of reasoning. This follows directly from the decidability results for the extension with Boolean constructs on atomic and inverse atomic roles (captured by $\mathcal{DLR}$). Indeed, $R_1 \subseteq R_2$ is equivalent to $\forall(R_1 \sqcap \neg R_2).\bot$, and thus can be expressed using difference of roles. We observe also that *universal* and *existential role agreements* introduced in [Han92], which allow one to define concepts by posing various types of constraints that relate the sets of fillers of two roles, can be expressed by means of intersection and difference of roles. Thus reasoning in the presence of role agreements is decidable, provided these constructs are applied only to basic roles.

2.13.3 Number restrictions on complex roles

In $\mathcal{ALCFI}_{reg}$, the use of (qualified) number restrictions is restricted to atomic and inverse atomic roles, which guarantees that the logic has the tree model property. This property is lost, together with decidability, if functional restrictions may be imposed on arbitrary roles. The reduction to show undecidability is analogous to the one used for intersection of roles, except that now functionality of a complex role (i.e., $\leqslant 1\,(\mathsf{right} \circ \mathsf{up}) \sqcup (\mathsf{up} \circ \mathsf{right})$) is used instead of role intersection to define the grid.

An example of decidable logic that does not have the tree model property is obtained by allowing the use of role composition (but not transitive closure) inside number restrictions. Let us denote with $\mathcal{N}(X)$, where X is a subset of $\{\sqcup, \sqcap, \circ, ^- \}$, unqualified number restrictions on roles that are obtained by applying the role constructs in X to atomic roles. Let us denote with $\mathcal{ALCN}(X)$ the description logic obtained by extending $\mathcal{ALC}$ with number restrictions in $\mathcal{N}(X)$. As shown in [BS99], concept satisfiability is decidable for the logic $\mathcal{ALCN}(\circ)$, even when extended with number restrictions on union and intersection of role chains of the same length. Notice that, decidability for $\mathcal{ALCN}(\circ)$ holds only for reasoning on concept expressions and is lost if one considers reasoning with respect to a TBox (or alternatively adds transitive closure of roles) [BS99]. Reasoning even

with respect to the empty TBox is undecidable if one adds to $\mathcal{ALCN}$ number restrictions on more complex roles. In particular, this holds for $\mathcal{ALCN}(\sqcap, \circ)$ (if no constraints on the lengths of the role chains are imposed) and for $\mathcal{ALCN}(\sqcup, \circ, ^-)$ [BS99]. The reductions exploit again the tiling problem, and make use of number restrictions on complex roles to simulate a universal role that is used for imposing local conditions on all points of the grid.

Summing up we can state that the borderline between decidability and undecidability of reasoning in the presence of number restrictions on complex roles has been traced quite precisely, although there are still some open problems. E.g., it is not known whether concept satisfiability in $\mathcal{ALCN}(\sqcup, \circ)$ is decidable (although logical implication is undecidable) [BS99].

3 Expressive Description Logics: The $\mathcal{SH}$ Family

The $\mathcal{SH}$ family of description logic is inspired by demands from applications in the sense that the language facilities of description logics of this family are designed in such a way that (i) application requirements w.r.t. expressivity can be fulfilled and (ii) reasoners can be built that are efficient in the average case. As we will see in this section, work on ontology languages based on the $\mathcal{SH}$ family revealed that there are intricate interactions between language constructs required for practical applications. Thus, reasoning gets harder (from a theoretical and from a practical point of view) if the expressivity is increased. We start our discussion of expressive description logics of the $\mathcal{SH}$ family with the core logic $\mathcal{ALC}$.

3.1 The foundation: $\mathcal{ALC}$

The well-known description logic $\mathcal{ALC}$ [BN03] is the backbone of all logics of the $\mathcal{SH}$ family. Let A be a concept name and R be a role name. Then, the set of $\mathcal{ALC}$ concepts (denoted by C or D) is inductively defined as follows:

$$C, D \rightarrow A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \forall R.C \mid \exists R.C$$

The semantics is defined in the standard way: given an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$:

$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}, R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$ (complement)

$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$ (conjunction)

$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$ (disjunction)

$(\exists R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \exists y.\, (x, y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$ (existential restriction)

$(\forall R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \forall y.\, (x, y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$ (value restriction)

As we have seen in the previous section, we say the a concept C is satisfiable if there exists an interpretation such that $C^{\mathcal{I}} \neq \emptyset$. The concept satisfiability problem of the logic $\mathcal{ALC}$ was shown to be PSPACE-complete in [SSS91]. For $\mathcal{ALC}$ the finite model property holds.

A TBox is a set of axioms of the form $C \sqsubseteq D$ (also called generalized concept inclusion, or GCI). A GCI $C \sqsubseteq D$ is satisfied by an interpretation $\mathcal{I}$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. An interpretation which satisfies all axioms of a TBox is called a model of the TBox. A concept C is satisfiable w.r.t. a TBox $\mathcal{T}$ if there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}} \neq \emptyset$. A concept D subsumes a concept C w.r.t. a TBox if for all models $\mathcal{I}$ of the TBox it holds that $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. D is called the subsumer, C is the subsumee. A concept name A_1 mentioned in a TBox is called a most-specific subsumer of a concept name A_2 (mentioned in the TBox and different from A_1) if A_1 subsumes A_2 and there is no other concept name A_3 (mentioned in the TBox and different from A_1 and A_2) such that A_1 subsumes A_3 and A_3 subsumes A_2. The least general subsumee of a concept name is defined analogously. The classification problem for a TBox is to find the set of most-specific subsumers of every concept name mentioned in the TBox (or knowledge base). The induced graph is called the subsumption hierarchy of the TBox.

The problem of verifying satisfiability or checking subsumption w.r.t. generalized TBoxes is ExpTime-hard [Neb90, Lut99a] (in fact ExpTime-complete – cf. Section 2.4). However, this result holds for the worst case, which does not necessarily occur in practical applications (see, e.g., [SvRvdVM95]), and practical work on reasoners for languages of the $\mathcal{SH}$ family exploits this insight.

As seen in above, an ABox is a set of assertions of the form $C(a)$, $R(a, b)$, $a = b$, or $a \neq b$. An ABox assertion is satisfied by an interpretation $\mathcal{I}$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$, $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$, $a^{\mathcal{I}} = b^{\mathcal{I}}$, and $a^{\mathcal{I}} \neq b^{\mathcal{I}}$, respectively. The ABox satisfiability problem (w.r.t. a TBox) is to check whether there exists an interpretation (a model of the TBox) that satisfies all ABox assertions. As usual, we define a knowledge base (KB) to be a pair $(\mathcal{T}, \mathcal{A})$ of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. A model of a KB is an interpretation that is a model of $\mathcal{T}$ and $\mathcal{A}$. The instance problem $instance_{(\mathcal{T},\mathcal{A})}(i, C)$ w.r.t. a knowledge base $(\mathcal{T}, \mathcal{A})$ is to test whether $i^{\mathcal{I}} \in C^{\mathcal{I}}$ for all models of the knowledge base. We say $instance_{(\mathcal{T},\mathcal{A})}(i, C)$ is entailed. The knowledge base is often omitted in the notation if clear from context. The instance retrieval problem $retrieve_{(\mathcal{T},\mathcal{A})}(C)$ w.r.t. a KB $(\mathcal{T}, \mathcal{A})$ and a query concept C is to determine all individuals i mentioned in the ABox for which $instance(i, C)$ is entailed. A role filler for a role R w.r.t. an individual i is an individual j (mentioned in the ABox) such that for all models $\mathcal{I}$ it holds that $(i^{\mathcal{I}}, j^{\mathcal{I}}) \in R^{\mathcal{I}}$ (we say $related(i, j, R)$ is entailed).

The inference problems mentioned in this section are called standard inference problems for TBoxes and ABoxes, respectively. Reasoners of the $\mathcal{SH}$ family support standard inference problems either for TBoxes and ABoxes or for TBoxes only. As we have seen, $\mathcal{ALC}$ inference problems are not tractable in the worst case, and, at the beginning, incomplete algorithms were used in concrete system implementations for expressive DLs. However, at the end of the eighties it became clear that incomplete algorithms for expressive description logics cause all kinds of problems for applications. For instance, more often than not, the addition of an axiom or assertion to the knowledge base led to the situation that previously obtained entailments were no longer computed due to peculiarities of the inference algorithm.

The beginning of the $\mathcal{SH}$ family started with work on the system KRIS [BH91b, HLPT91, AHLM91], which provides a sound and complete reasoner based on the tableau calculus presented in [SSS91]. KRIS supports $\mathcal{ALC}$ plus number restrictions (plus some additional language constructs). The KRIS reasoner implements optimization techniques for the concept and ABox satisfiability problem w.r.t. TBoxes (e.g., lazy unfolding, trace technique). The main achievement of this work is that the architecture of KRIS is tailored towards specific services for TBoxes, namely TBox classification. Specific optimization techniques for the classification problem developed for KRIS are used by all contemporary reasoners of the $\mathcal{SH}$ family (see below). The idea is to classify a TBox using a top-down and bottom-up search phase for computing the most-specific subsumers and least-specific subsumees based on subsumption tests. KRIS avoids unnecessary subsumption tests using marker propagation tests [BHN+92, BFH+94].

3.2 Concrete domains

Another achievement of the work on description logics that is also important for ontology languages is the treatment of specific domains with fixed (concrete) semantics. Applications require constraints over the reals, the integers, or a domain of strings. A concrete domain $\mathcal{D}$ is a tuple $(\Delta^{\mathcal{D}}, \Phi)$ of a non-empty set $\Delta^{\mathcal{D}}$ and a set of predicates Φ. Predicates are defined in a certain language (e.g., linear inequations over polynomials or equations over strings). The integration of concrete domains into $\mathcal{ALC}$ is investigated in [BH91a, BH92]. The idea of the new language, $\mathcal{ALC}(\mathcal{D})$, is that the axioms for capturing the concrete semantics of the objects in $\Delta^{\mathcal{D}}$ is not captured with description logic axioms but somehow represented separately. The tableau calculus in [BH91a, BH92] treats the satisfiability problem w.r.t. to conjunctions of concrete domain predicates as separate subproblems. The concrete domain satisfiability problems must be decidable (admissibility criterion for concrete domains). A Tableau Algorithm for Description Logics with Concrete Domains is also provided in [LM05].

With concrete domains, so-called attributes are introduced, which are partial functions that map individuals of the abstract domain $\Delta^{\mathcal{I}}$ to elements of $\Delta^{\mathcal{D}}$ of the concrete domain $\mathcal{D}$. For attributes a, the interpretation is extended as follows: $a^{\mathcal{I}} : \Delta^{\mathcal{I}} \longrightarrow \Delta^{\mathcal{D}}$.

It is important to note that in the original approach [BH91a, BH92] it is possible to relate (multiple) attribute values of *different* individuals of the domain $\mathcal{I}$. One can represent, for instance, structures such as lists of numbers with decreasing value where each value is at most half as large as the predecessor. If the language provides concrete domains such as, for instance, linear inequations over the reals, GCIs cannot be supported by description logic part due to undecidability of major inference problems. This follows from a result in [Lut04] (a direct proof was developed at the same time and is given in [Möl00]). In a restricted form where no feature compositions can be used, it is only possible to relate attribute values of a single element of $\mathcal{I}$. We use the notation $\mathcal{ALC}(\mathcal{D})^-$ to indicate that feature chains are omitted. Concrete domains are part of many specific description logics of the $\mathcal{SH}$ family that we cover below.

3.3 Transitive roles

For many applications, part-whole relations are important. A characteristic of some part-whole relations is that they are transitive (see, e.g., [Lam96]). In order to cope with these modeling demands, for instance, in process engineering applications, an investigation about an extensions of $\mathcal{ALC}$ with means to express transitivity was carried out [Sat96, Sat98]. $\mathcal{ALC}$ was extended with a transitive closure operator, with transitive roles, and with so-called transitive orbits. As discussed in other sections, $\mathcal{ALC}$ extended with a transitive closure operator causes the concept satisfiability problem to move from PSPACE to ExpTime.

Syntactically, transitive roles are indicated as a subset of all role names. It turned out that $\mathcal{ALC}$ extended with transitive roles remains in PSPACE [Sat96]. Transitive roles have the semantics that for all transitive roles R the models must satisfy $R^{\mathcal{I}} = (R^{\mathcal{I}})^{+}$. Thus, transitive roles are "globally" transitive and cannot be used in a transitive way in a local setting only (as possible with a specific operator for the transitive closure of a role).

Inspired by work on modal logics, [Sat96] introduces an elegant way to integrate reasoning about transitive roles into the $\mathcal{ALC}$ tableau calculus by a special rule for transitive roles in value restrictions. Additionally, in order to enforce termination, *blocking conditions* were defined such that the calculus terminates. A blocking condition involves a test whether two sets of concepts are in a certain relation (for $\mathcal{ALC}_{R^+}$, the relation is $\subseteq$, for details see [Sat96]).

The logic was initially called $\mathcal{ALC}_{R^+}$. As more language constructs were added later on, and acronyms became hard to read, $\mathcal{ALC}_{R^+}$ was renamed $\mathcal{S}$ in [HST99].[7]

3.4 Role hierarchies and functional restrictions

Inspired by work on medical domains in which it became important to represent that some relations are subrelations (subsets) of other relations, so-called role inclusions axioms of the form $R \sqsubseteq S$ (with R and S being role names) were investigated in [Hor97] as an extension to $\mathcal{ALC}_{R^+}$. A set of role inclusion axioms is called a role hierarchy. Models for role hierarchies are restricted to satisfy $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ for all $R \sqsubseteq S$. The description logic is called $\mathcal{ALCH}_{R^+}$ or $\mathcal{SH}$.

Role hierarchies introduce explicit names for so-called subroles. In [Hor98b] it is argued that role hierarchies provide for adequate expressivity while still allowing for efficient practical implementations at the same time. Another possibility would have been to consider a role-forming operator for constructing role conjunctions ($R \sqcap S$). However, except for inverse roles (see below) the $\mathcal{SH}$ family includes no role-forming operators in order to provide for practically efficient implementations (see also the discussion about a transitive closure operator for roles in the previous subsection).

Additionally, in [Hor97, Hor98b] global functional restrictions on roles were investigated. In the corresponding description logic $\mathcal{ALCH}f_{R^+}$ so-called features were introduced as a specific subset of the role names.[8] Features must not be transitive. The semantics

[7] The name is inspired by modal logic $\mathbf{S4_m}$ but, obviously, it is a misnomer. However the name is kept for historical reasons.

[8] Note that $\mathcal{ALCH}f_{R^+}$ does not provide role-value maps as supported by $\mathcal{ALCF}$ [HN90, Hol94].

of a feature f is a (single valued) partial function $f^{\mathcal{I}} : \Delta^{\mathcal{I}} \longrightarrow \Delta^{\mathcal{I}}$.

With several examples, the interactions of role hierarchies and functional restrictions on roles were demonstrated in [Hor97]. A sound and complete tableau calculus for $\mathcal{ALCH}f_{R^+}$ is described in [Hor98c]. This tableau calculus provided the basis for the enormous success of the $\mathcal{SH}$ family of ontology languages. Based on an optimized implementation of this calculus in the system FaCT [Hor97, Hor98b] it was shown that description logics could provide a solid basis for practical application of ontology languages. Role hierarchies and transitive roles allow one to somehow "simulate" GCIs (by constructing an equisatisfiable knowledge base). However, the FaCT system also included full support for GCIs.

The contribution of the $\mathcal{ALCH}f_{R^+}$ reasoner FaCT is (atleast) threefold. First, improvements to propositional satisfiability search algorithms [Fre95] were incorporated into description logic systems (backjumping, boolean constraint propagation, semantic branching, etc.) and, second, classification operations were dramatically increased by the invention of a so-called model merging operation [Hor97], which exploits that most subsumption tests for concept names A_1 and A_2 used to compute the subsumption hierarchy return false. The idea of a model merging operation is to compute (small) data structures for concept names (and their negations) such that it is more or less directly obvious that the conjunction $A_1 \sqcap \neg A_2$ is satisfiable (i.e., there is no subsumption relation). Third, using algebraic transformation, FaCT showed that, in many practical applications, corresponding TBox axioms can be converted into a form such that lazy unfolding is maximally exploited in the tableau calculus (GCI absorption [HT00b]). The system FaCT initiated the breakthrough of description logics as the basis for practically used ontology languages. FaCT was designed for TBoxes only.

3.5 Number restrictions and inverse roles

In particular for technical applications the need for restrictions on the number of role fillers of an individual became apparent. Number restrictions are concept construction operators of the form $(\leq n\, R)$ or $(\geq n\, R)$ (simple number restrictions, indicated with letter $\mathcal{N}$ in language names) and $(\leq n\, R.C)$ or $(\geq n\, R.C)$ (qualified number restrictions [HB91], indicated with letter $\mathcal{Q}$ in language names). For simple number restrictions, interpretations must satisfy $(\leq n\, R)^{\mathcal{I}} = \{x \,|\, \#\{y|(x,y) \in R^{\mathcal{I}}\} \leq n\}$ and $(\geq n\, R)^{\mathcal{I}} = \{x \,|\, \#\{y|(x,y) \in R^{\mathcal{I}}\} \geq n\}$. For qualified number restrictions, interpretations must satisfy $(\leq n\, R.C)^{\mathcal{I}} = \{x \,|\, \#\{y|(x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \leq n\}$ and $(\geq n\, R.C)^{\mathcal{I}} = \{x \,|\, \#\{y|(x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \geq n\}$.

KRIS supported simple number restrictions in a system implementation at the end of the eighties. With only simple number restrictions and no role inclusions, it is possible to use a single placeholder in the tableau calculus for an arbitrary large number of role fillers required by a number restrictions. Results on the interaction of number restrictions and role conjunctions were developed with $\mathcal{ALCNR}$ [BDS93b, BDS93a]. Simple reasoning with placeholder is no longer possible. The same holds for number restrictions in combination with role hierarchies as used in the $\mathcal{SH}$ family.

In addition to problems w.r.t. placeholder reasoning in the presence of number restrictions, it was shown that there is a strong interaction between number restrictions and

transitive roles (and role hierarchies). Allowing number restrictions with transitive roles (or roles which have transitive subroles) leads to undecidability [HST99]. As a consequence, so-called *simple* roles were introduced into the $\mathcal{SH}$ family. In (qualified) number restrictions, only simple roles are allowed. With this restriction, inference problems become decidable [HST99].

Another demand from practical applications was the support for inverse roles (letter $\mathcal{I}$ in language names). In [HST99] the research on a corresponding role-forming operator $\cdot^{-1}$ in the context of the $\mathcal{SH}$ family is summarized. Again, a subtle interaction between number restrictions (or features), inverse roles as well as transitive roles and role hierarchies (or GCIs) was discovered. If all these constructs are used, the finite model property does no longer hold. First, due to inverse roles, the trace technique is no longer applicable, and, second, the application of the blocking condition introduced in the work about $\mathcal{ALC}_{R^+}$ had to be made considerably more complex. Blocking must be dynamic [HST99]. This makes the implementation of concrete reasoning systems much more difficult. An additional source of complexity is that blocking must not occur too early (thus, the blocking condition involves a test for set equality), and, furthermore, due to infinite models, the blocking condition involves a pair of two sets of concepts (pairwise blocking [HST99]).

Although ABoxes were also investigated in the context of $\mathcal{ALCNR}$, work on the $\mathcal{SH}$ family of description logic languages initially considered TBoxes only.

3.6 Number restrictions, ABoxes, and concrete domains

Inspired by work on the $\mathcal{SH}$ family and work on ABoxes in $\mathcal{ALCNR}$ as well as work on concrete domains [BH91a], a tableau calculus for ABox reasoning in the language $\mathcal{ALCNH}_{R^+}$ was presented in [HM00] and concrete domain reasoning was investigated in this context in [HMW01]. The insights of this work are that in the presence of ABoxes, (i) models are no longer (quasi) trees but forests, (ii) individuals mentioned in the ABox must not block each another, and (iii) on the concrete domain part of the language, feature chains cannot be supported (for all kinds of concrete domains) in order to preserve decidability. A tableau calculus for reasoning about ABoxes in the language $\mathcal{SHIQ}$ (aka $\mathcal{ALCQHI}_{R^+}$) with ABoxes was presented shortly afterwards in [HST00b] (but concrete domains were not considered).

The latter work led to a new version of the FaCT system (iFaCT) for supporting TBoxes with inverse roles. The above-mentioned research contributions also provided the basis for the implementation of the RACER reasoner [HM01c], a DL system for TBoxes and ABoxes with concrete domains for linear inequations over the reals and the cardinals as well as inequations over strings and booleans. First versions of both systems, iFaCT and RACER, appeared at the end of the nineties, i.e. both systems support the language $\mathcal{SHIQ}$.[9] Both RACER and FaCT use the TBox classification techniques developed for KRIS [BHN+92, BFH+94].

Optimized reasoning techniques for $\mathcal{SHIQ}$ w.r.t. blocking [HS02] were developed for later versions of the iFaCT system, and also included in RACER. The idea is to relax the

[9] In RACER initially the unique name assumption was always employed, in later versions the assumption could be activated on demand.

blocking condition for inverse roles (see above) and retain the subset tests for some parts of the concept set involved in the blocking test (see [HS02] for details).

With RACER, optimized reasoning for qualitative number restrictions [HM01b, HM01a] was investigated. The work is based on [OK99]. In order to classify huge terminologies with RACER, a refinement of the techniques introduced in [BHN+92, BFH+94] is presented in [HM01c]. Topological sorting of transformed GCIs to classify concepts in definition order allows to skip the bottom-up search phase. Optimizations for concrete domains in terms of extended model merging operations and incremental concrete domain satisfiability testing during a tableau proof are described in [HMT01]. GCI absorption strategies are also investigated with RACER, e.g., absorption of domain and range restrictions (see also [TH04] for similar techniques in FACT).

Reasoning systems for the $\mathcal{SH}$ family are successful because of research on average-case behavior and appropriate optimization techniques. Systems analyze the language of the input problem and select appropriate optimizations to answer queries as fast as possible, moreover, they are based on sound and complete algorithms.

Optimizations for instance retrieval w.r.t. ABoxes is investigated in [HM04]. An important property of the $\mathcal{SHIQ}$ language is that the subsumption hierarchy of the TBox part of a knowledge base $(\mathcal{T}, \mathcal{A})$ is stable w.r.t. additions to the ABox part. Stability means that the subsumption relation between concepts C and D depends only on axioms in $\mathcal{T}$. This property is exploited in practical ABox systems such as RACER (and also older systems such as KRIS). Multiple knowledge bases $(\mathcal{T}, \mathcal{A}_1)$, ..., $(\mathcal{T}, \mathcal{A}_k)$ with the same TBox $\mathcal{T}$ can be efficiently supported in the sense that computations for the TBox can be reused for answering queries on any of the ABoxes $\mathcal{A}_i$. Unfortunately, the stability property is lost with the introduction of cardinalities for concepts or with the inclusion of so-called nominals, which became necessary in order to further increase the expressivity of $\mathcal{SHIQ}$ for some applications.

3.7 Nominals

A *nominal* (cf. Section2.7) denotes a singleton concept. The syntax is $\{o\}$ and the semantics w.r.t. the interpretation is $\{o\}^{\mathcal{I}} = \{o^{\mathcal{I}}\}$. With nominals it is possible to relate all individuals of a certain concept to a particular individual (e.g., all humans stem from a particular human called Adam). The first system with support for nominals was CRACK [BFT95].

Although nominals in the context of $\mathcal{SHIQ}$ were proven to be decidable (see [Tob01]) it took some time until the first tableau calculus was presented for the language $\mathcal{SHOQ}(\mathcal{D})$ [HS01]. This work also introduced so-called datatype roles [HS01], which must not be confused with concrete domain attributes. Datatype roles map object from the domain to *sets of* objects from a concrete domain. In $\mathcal{SHOQ}(\mathcal{D})$ concrete domain predicates apply to (multiple) datatype properties of single object of the interpretation domain $\Delta^{\mathcal{I}}$. It is *not* possible to enforce constraint on datatype values of *multiple* objects from $\Delta^{\mathcal{I}}$. The insight gives rise for corresponding optimization techniques but it should be noted that some expressivity is lost.

The distinction between TBoxes and ABoxes is no longer required for languages with nominals. Instead of using $C(a)$ or $R(a, b)$ as ABox assertion one can just write GCIs

such as $\{a\} \sqsubseteq C$ or $\{a\} \sqsubseteq \exists R.\{b\}$, respectively.[10] Even if ABoxes would be supported by practical systems, it is obvious that the subsumption relation is not stable for languages with nominals.

Intricate interactions of nominals in $\mathcal{SHOQ}(\mathcal{D})$ with inverse roles were investigated in $\mathcal{SHOIQ}(\mathcal{D})$ [HS05]. Indeed, it was shown that concept satisfiability in $\mathcal{SHOIQ}(\mathcal{D})$ is NExpTime-complete.

3.8 The research frontier of the $\mathcal{SH}$ family

Further results on optimized classification [TH05b, TH05a] has opened up additional application areas for ontology languages. And, although much has been achieved by dedicated optimization techniques developed for the $\mathcal{SH}$ family of description logic languages, still there are hard knowledge bases known (e.g., [BCG01]). New languages features with respect to specific kinds of role axioms involving role composition have been proposed for medical domains. A tableau calculus for the new language $\mathcal{SROIQ}(\mathcal{D})$ is presented in [HKS06]. To the best of our knowledge, there is no system implementation at the time of this writing, that supports all features of this language.

Concrete domain reasoning is also actively explored. Starting with investigation involving new combination operators ([Lut99b]), in [Lut01a, Lut01b] it is shown that for specific concrete domains, feature chains can indeed be allowed in the presence of GCIs (see also [Lut03, Lut04]). The language investigated (Q-$\mathcal{SHIQ}$) provides predicates for linear inequalities between variables (but no polynoms). A more modular approach is described in [LM05, LLMW06] where the notion of admissibility (see above) is extended to a so-called ω-admissibility. No system implementation exists at the time of this writing.

New versions of the description logic systems discussed in the previous section have been developed. These systems are FaCT++ (for $\mathcal{SHOIQ}(\mathcal{D})$) [TH06] and RacerPro (at the time of this writing the latter only provides an approximation for nominals). FaCT++ is written in C++ whereas RacerPro is implemented in CommonLisp. A new Java-based description logic system for $\mathcal{SHOIQ}(\mathcal{D})$ (and OWL DL) is Pellet. As FaCT++, Pellet is based on a tableau reasoning algorithm and integrates various optimization techniques in order to provide for a fast and efficient practical implementation. New developments also tackle the problem of "repairing" knowledge bases in case an inconsistency is detected [KPSH05]. In addition, with Pellet, optimization techniques, for instance, for nominals have been investigated [SGP06]. Other description logic systems are described in [MH03].

Compared to initial approaches for query languages (see [LS91]), recently, more expressive languages for instance retrieval have been investigated (conjunctive queries [CDGL98a, HT00a, GH05]). To the best of the authors' knowledge, algorithms for answering conjunctive queries for expressive description logics such as $\mathcal{SHIQ}$ are not known. In practical systems such as Racer implementations for a restricted form of conjunctive queries is available (variables are bound to individuals mentioned in the ABox only). Database-inspired optimization techniques for this language in the context of a tableau prover are presented in [MHW06]. In addition, Racer supports the incremental computation of result sets for restricted conjunctive queries. The demand for efficient instance

[10] Equality and inequality of individuals can also easily be specified using negation.

retrieval has led to the development of new proof technique for languages of the $\mathcal{SH}$ family. A transformation approach using disjunctive datalog [EGM97], resolution techniques as well as magic-set transformations to support reasoning for $\mathcal{SHIQ}$ is described in [HMS04, Mot06] with encouraging results. In this context, a new system, KAON2 has demonstrated that techniques from the database community can be successfully used also for implementing description logic systems. Although at the time of this writing, KAON2 is a very recent development and not quite as expressive as FACT++, PELLET, or RACERPRO (e.g., w.r.t. datatypes, nominals, large numbers in qualified number restrictions, etc.).

Recent advances in the development of standards for description logic reasoning systems (such as DIG [BHMC03]) have contributed to the fact that DL systems can be interchanged such that the strength of particular reasoning systems can be exploited for building practical applications. Since semantic web applications have become interesting from a business point of view, commercial DL systems have appeared (e.g., CEREBRASERVER from Cerebra Inc.) and commercial versions of above-mentioned systems became available (e.g., KAON2 from Ontoprise or RACERPRO from Racer-Systems and Franz Inc.).

4 Tractable Description Logics

In this section we concentrate on those families of description logics for which the main reasoning problems can be decided in PTIME (or less – see later).

The quest for tractable (i.e., PTIME decidable) description logics (DLs) started in the 1980s after the first intractability results for DLs were shown [BL84, Neb88]. Until recently, it was restricted to DLs that extend the basic language $\mathcal{FL}_0$, which comprises the concept constructors conjunction ($\sqcap$) and value restriction ($\forall r.C$). The main reason for this focusing was that, when clarifying the logical status of property arcs in semantic networks and slots in frames (which are the ancestors of modern DLs), the decision was taken that arcs/slots should be read as value restrictions rather than existential restrictions ($\exists r.C$).

In almost every application of DLs, it is crucial to reason with terminologies (also called TBoxes or DL ontologies), rather than with isolated concept descriptions. Unfortunately, as soon as TBoxes were taken into consideration, tractability turned out to be unattainable in $\mathcal{FL}_0$: even classifying the simplest form of TBoxes that admit only acyclic concept definitions was shown to be coNP-hard [Neb90]. If the most general form of TBoxes is admitted, which consists of general concept inclusion axioms (GCIs) as supported by all modern DL systems, then classification in $\mathcal{FL}_0$ even becomes EXPTIME-complete [BBL05].

For these reasons, and also because of the need for expressive DLs in applications, from the mid 1990s on, the DL community has mainly given up on the quest of finding tractable DLs. Instead, it investigated more and more expressive DLs, for which reasoning is worst-case intractable. The goal was then to find practical reasoning procedures, i.e., algorithms that are easy to implement and optimize, and which—though worst-case exponential or even worse—behave well in practice (see, e.g., [HST00a]). This line of research has

resulted in the availability of highly optimized DL systems for expressive DLs based on tableau algorithms [Hor98a, HM01c], and successful applications: most notably the recommendation by the W3C of the DL-based language OWL [HPSvH03] as the ontology language for the Semantic Web.

Recently, the choice of value restrictions as a sine qua non of DLs has been reconsidered. On the one hand, it was shown that the DL $\mathcal{EL}$, which allows for conjunction ($\sqcap$) and existential restrictions ($\exists r.C$), has better algorithmic properties than $\mathcal{FL}_0$. Subsumption of both acyclic and cyclic $\mathcal{EL}$ TBoxes is tractable [Baa03a], and this remains so even if general TBoxes with GCIs are admitted [Bra04]. On the other hand, there are applications where value restrictions are not needed, and where the expressive power of $\mathcal{EL}$ or small extensions thereof appear to be sufficient. In fact, the Systematized Nomenclature of Medicine, employs $\mathcal{EL}$ with an acyclic TBox [CRP+93, Spa00]. The Gene Ontology [The00] can be seen as an acyclic $\mathcal{EL}$ TBox with one transitive role. Finally, large parts of the Galen Medical Knowledge Base can also be expressed in $\mathcal{EL}$ with GCIs, role hierarchy, and transitive roles [RH97].

This characteristic of expressive description logics makes their usage particularly unsuited in all such contexts in which Another strong reason for insisting on tractability stems from the need of using (DLs) ontologies a conceptual view over data repositories. This is the case, for example, of Enterprise Application Integration Systems, Data Integration Systems [Len02], or the Semantic Web [HH01], where data become instances of concepts in ontologies. In these contexts, data are typically very large and dominate the intentional level of the ontologies. Hence, when measuring the computational complexity of reasoning, the most important parameter is the size of the data, i.e., one is particularly interested in *data complexity* [Var82]. While in all the above mentioned contexts one could still accept reasoning that is exponential on the intentional part, it is mandatory that reasoning is polynomial in the data. A second fundamental requirement is the possibility to answer queries over an ontology that are more complex than the simple queries (i.e., concepts and roles) usually considered in Description Logics (DLs) research.

Traditionally, research carried out in DLs has not paid much attention to data complexity, and only recently efficient management of large amounts of data has become a primary concern in ontology reasoning systems [HLTB04, CHW05]. Unfortunately, research on the trade-off between expressive power and computational complexity of reasoning has shown that many DLs with efficient, i.e., worst-case polynomial time, reasoning algorithms lack the modeling power required for capturing conceptual models (such as UML class diagrams and Entity-Relationship diagrams) and basic ontology languages. On the other hand, whenever the complexity of reasoning is exponential in the size of the instances, there is little hope for effective instance management.

In the rest of this section we provide a specification of different families of DLs for which the main reasoning problems can be decided in PTime.

Among DLs reasoning services, when studying such DLs we are particularly interested in those problems for which tractability w.r.t. data complexity turns out to be crucial, whether intractability in the size of the TBox or the query in input (e.g., for query answering) is instead in general acceptable. Therefore, besides classical DLs reasoning services, as logical implication or subsumption of concepts and roles, we also consider here, instance retrieval, and query answering (formally defined below) in different families

of tractable DLs, which differ in terms of expressivity and kinds of applications towards which they are designed. As for expressive power, a desirable property is that a tractable DL should include the main modeling features of conceptual models, which are also at the base of most ontology languages. These features include cyclic assertions, ISA on concepts, inverses on roles, role typing, mandatory participation to roles, and functional restrictions of roles. We also look at query languages that go beyond the expressive capabilities of concept expressions in DLs, in particular we focus on conjunctive queries (corresponding to the select-project-join fragment of SQL). Formally, a conjunctive query (CQ) q over a knowledge base $\mathcal{K}$ is an expression of the form

$$q(\vec{x}) \leftarrow \exists \vec{y}.\, conj(\vec{x}, \vec{y})$$

where $\vec{x}$ are the so-called *distinguished variables*, $\vec{y}$ are existentially quantified variables called the *non-distinguished* variables, and $conj(\vec{x}, \vec{y})$ is a conjunction of atoms (and possibly equalities) with free variables $\vec{x}$ and $\vec{y}$, expressed in terms of concepts and roles of $\mathcal{K}$. A query $q(\vec{x}) \leftarrow \exists \vec{y}.\, conj(\vec{x}, \vec{y})$ is interpreted over an interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ as the set $q^{\mathcal{I}}$ of tuples $\vec{o} \in \Delta^{\mathcal{I}} \times \cdots \times \Delta^{\mathcal{I}}$ such that, when we assign the objects $\vec{o}$ to the variables $\vec{x}$, the formula $\exists \vec{y}.\, conj(\vec{x}, \vec{y})$ evaluates to true in $\mathcal{I}$.

Formally, the reasoning services we are interested in are:

- *instance retrieval*, i.e., given a KB $\mathcal{K}$, a concept C (resp., a role R), return the set of constants a (resp. the set of pairs $\langle a, b \rangle$ of constants) of $\mathcal{K}$, such that in every model $\mathcal{I}$ of $\mathcal{K}$ we have $a^{\mathcal{I}} \in C^{\mathcal{I}}$ (resp. $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$).

- *query answering*: given a query q with distinguished variables $\vec{x}$ and a KB $\mathcal{K}$, return the set $ans(q, \mathcal{K})$ of tuples $\vec{c}$ of constants of $\mathcal{K}$ such that in every model $\mathcal{I}$ of $\mathcal{K}$ we have $\vec{c}^{\mathcal{I}} \in q^{\mathcal{I}}$.

We point out that the decision problem associated to instance retrieval, is the so-called *instance checking*, i.e., given a KB $\mathcal{K}$, a concept C and a constant a (resp., a role R and a pair of constants a and b), verify whether in every model $\mathcal{I}$ of $\mathcal{K}$ we have $a^{\mathcal{I}} \in C^{\mathcal{I}}$ (resp. $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$).

As for query answering, the associated decision problem, called *recognition problem for query answering*, is as follows: given a KB $\mathcal{K}$, a query q, and a tuple of constants $\vec{c}$ of $\mathcal{K}$, check whether $\vec{c} \in ans(q, \mathcal{K})$. When we talk about the computational complexity of instance retrieval or query answering, in fact, we implicitly refer to the associated decision problem.

Note that, query answering is a generalization of instance retrieval (actually, instance retrieval can be seen as a query answering problem in which queries ask for the extension of single concepts or single roles).

In the following we first present the DL *DL-Lite*, then we move to $\mathcal{EL}^{++}$ and extension of $\mathcal{EL}$, finally we briefly review DLP and Horn-$\mathcal{SHIQ}$.

4.1 *DL-Lite*

DL-Lite is a description logics specifically tailored for those contexts where ontologies are used to access large amounts of data. Informally, *DL-Lite* is a fragment of OWL-DL

which allows for capturing conceptual modeling constructs, while keeping query answering efficient. Specifically, efficiency of query answering should be achieved by delegating data storage and query answering to an RDBMS.

The distinguishing features of *DL-Lite* are that the extensional component of a knowledge base, the ABox, is maintained by an RDBMS in secondary storage, and that query answering can be performed as a two step process: in the first step, a query posed over the knowledge base is reformulated, taking into account the intensional component (the TBox) only, obtaining a union of conjunctive queries; in the second step such a union is directly evaluated over the ABox, and the evaluation can be carried out by an SQL engine, taking advantage of well established query optimization strategies. Since the first step does not depend on the data, and the second step is the evaluation of a relational query over a databases, the whole query answering process is in LogSpace in the data [AHV95].

DL-Lite is essentially the maximal fragment exhibiting such a desirable property, and allowing one to delegate query evaluation to a relational engine [CDGL$^+$05b]. Indeed, even slight extensions of *DL-Lite* make query answering (actually already instance checking, i.e., answering atomic queries) at least NLogSpace in data complexity, ruling out the possibility that query evaluation could be performed by a relational engine.

Notably, this was one of the motivations behind several research works done on CLASSIC in the 80's [BBMR89b].

The logic

As usual in DLs, *DL-Lite* allows for representing the domain of interest in terms of concepts, denoting sets of objects, and roles, denoting binary relations between objects. In *DL-Lite* concepts and roles are defined as follows:

$$\begin{array}{rcl} B & \longrightarrow & A \mid \exists R \\ C & \longrightarrow & B \mid \neg B \\ R & \longrightarrow & P \mid P^- \end{array}$$

where A denotes an atomic concept, P an atomic role, and P^- the inverse of the atomic role P; B denotes a *basic concept* that can be either an atomic concept or a concept of the form $\exists R$, the standard DL construct of unqualified existential quantification on roles (and their inverse); finally, C denotes a (general) *concept*, which can be a basic concept or its negation, whereas E denotes a (general) *role*, which can be a basic role or its negation.

A DL *knowledge base* (KB) $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ represents the domain of interest in terms of two parts, a *TBox* $\mathcal{T}$, specifying the intensional knowledge, and an *ABox* $\mathcal{A}$, specifying extensional knowledge. *DL-Lite* TBox assertions are of the form

$B \sqsubseteq C$	*inclusion assertion*
(funct R)	*functionality assertion*

An inclusion assertion expresses that a basic concept B is subsumed by a general concept C, while a functionality assertion expresses the (global) functionality of a role (atomic or inverse).

We observe that we might include $B_1 \sqcup B_2$ in the constructs for the left-hand side of the inclusion assertions (where $\sqcup$ denotes union) and $C_1 \sqcap C_2$ in the constructs for the right-hand side (where $\sqcap$ denotes conjunction). In this way, however, we would not extend

the expressive capabilities of the language, since these constructs can be simulated by considering that $B_1 \sqcup B_2 \sqsubseteq C$ is equivalent to the pair of assertions $B_1 \sqsubseteq C$ and $B_2 \sqsubseteq C$, and that $B \sqsubseteq C_1 \sqcap C_2$ is equivalent to $B \sqsubseteq C_1$ and $B \sqsubseteq C_2$. Similarly, we might add $\bot$ (denoting the empty set) to the constructs for the left-hand side and $\top$ (denoting the whole domain) to those for the right-hand side.

An ABox is formed by a set of *membership assertions* on atomic concepts and on atomic roles, of the form

$$A(a), \qquad P(a, b)$$

stating respectively that the object denoted by the constant a is an instance of A and that the pair of objects denoted by the pair of constants (a, b) is an instance of the role P.

As for the ABox, *DL-Lite* allows for assertions of the form:

$$B(a), \quad P(a, b) \qquad \textit{membership assertions}$$

where a and b are constants. These assertions state respectively that the object denoted by a is an instance of the basic concept B, and that the pair of objects denoted by (a, b) is an instance of the role P.

Although *DL-Lite* is quite simple from the language point of view, it allows for querying the extensional knowledge of a KB in a much more powerful way than usual DLs, in which only membership to a concept or to a role can be asked. Specifically, *DL-Lite* allows for using conjunctive queries of arbitrary complexity. A conjunctive query over a *DL-Lite* knowledge base $\mathcal{K}$ is a CQ in which atoms in the body are of the form $B(z)$, or $P(z_1, z_2)$, where B and R are respectively a basic concept and an atomic role in $\mathcal{K}$, and z, z_1, z_2 are constants in $\mathcal{K}$ or variables in $\vec{x}$ or $\vec{y}$.

An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a first order structure over $\Delta^{\mathcal{I}}$ with an *interpretation function* $\cdot^{\mathcal{I}}$ such that:

$$\begin{array}{rcl} A^{\mathcal{I}} & \subseteq & \Delta^{\mathcal{I}} \\ P^{\mathcal{I}} & \subseteq & \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \\ (P^-)^{\mathcal{I}} & = & \{(o_2, o_1) \mid (o_1, o_2) \in P^{\mathcal{I}}\} \\ (\exists R)^{\mathcal{I}} & = & \{o \mid \exists o' . (o, o') \in R^{\mathcal{I}}\} \\ (\neg B)^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \setminus B^{\mathcal{I}} \end{array}$$

An interpretation $\mathcal{I}$ is a *model* of an inclusion assertion $B \sqsubseteq C$ if and only if $B^{\mathcal{I}} \subseteq C^{\mathcal{I}}$; $\mathcal{I}$ is a model of a functionality assertion (funct R) if $(c, c') \in R^{\mathcal{I}}$ and $(c, c'') \in R^{\mathcal{I}}$ implies $c' = c''$; $\mathcal{I}$ is a model of a membership assertion $C(a)$ (resp., $R(a, b)$) if $a \in C^{\mathcal{I}}$ (resp., $(a, b) \in R^{\mathcal{I}}$). A *model of a KB* $\mathcal{K}$ is an interpretation $\mathcal{I}$ that is a model of all assertions in $\mathcal{K}$. A KB is *satisfiable* if it has at least one model. A KB $\mathcal{K}$ *logically implies* an assertion α if all the models of $\mathcal{K}$ are also models of α.

Expressivity of *DL-Lite*

Although equipped with advanced reasoning services, at first sight *DL-Lite* might seem rather weak in modeling intensional knowledge, and hence of limited use in practice. In fact, this is not the case. Despite the simplicity of its language and the specific form of

inclusion assertions allowed, *DL-Lite* is able to capture the main notions (though not all, obviously) of both ontologies, and of conceptual modeling formalisms used in databases and software engineering, such as Entity-Relationship diagrams and UML class diagrams. In particular, *DL-Lite* assertions allow us to specify:

- *ISA*, e.g., stating that a concept A_1 is subsumed by a concept A_2, using $A_1 \sqsubseteq A_2$;
- *disjointness*, e.g., between concepts A_1 and A_2, using $A_1 \sqsubseteq \neg A_2$;
- *role-typing*, e.g., stating that the first (resp., second) component of the role P is an instance of A, using $\exists P \sqsubseteq A$ (resp., $\exists P^- \sqsubseteq A$);
- *participation constraints*, e.g., stating that all instances of a concept A participate to a role P as the first (resp., second) component, using $A \sqsubseteq \exists P$ (resp., $A \sqsubseteq \exists P^-$);
- *non-participation constraints*, using $A \sqsubseteq \neg\exists R$;
- *functionality restrictions* on relations, using (funct R).

Observe that *DL-Lite* does allow for cyclic assertions without falling into intractability. Indeed, we can enforce the cyclic propagation of the existence of a P-successor using the two *DL-Lite* inclusion assertions $A \sqsubseteq \exists P$ and $\exists P^- \sqsubseteq A$. The constraint imposed on a model is similar to the one imposed by the $\mathcal{ALN}$ cyclic assertion $A \sqsubseteq \exists P \sqcap \forall P.A$, though stronger, since it additionally enforces the second component of P to be typed by A. In order to keep tractability even in the presence of cycles, *DL-Lite* imposes restrictions on the use of the $\forall R.C$ construct, which, if used together with inclusion assertions, immediately would lead to intractability of TBox reasoning [Cal96b] and of query answering.

Finally, notice that *DL-Lite* is a strict subset of OWL-Lite, the least expressive variant of OWL[11], which presents some constructs (e.g., forms of negation and disjunction) that cannot be expressed in *DL-Lite*, and that make reasoning in OWL-Lite non-tractable in general.

Reasoning in *DL-Lite*

We discuss now reasoning in *DL-Lite*, and concentrate on the basic reasoning task in the context of using ontologies to access large data repositories, namely that of answering (conjunctive) queries over a *DL-Lite* knowledge base. Obviously, since query answering generalizes instance retrieval, which in turn generalizes instance checking, solving it means solving also the other reasoning problems we are mainly interested in. Notice also that other classical forms of reasoning can be reduced to query answering [CDGL$^+$05b]. For example, to check whether $\mathcal{K}$ is unsatisfiable, we can simply add the inclusion $A_1 \sqcap A_2 \sqsubseteq \bot$ to the TBox and the assertion $A_1(a)$ to the ABox (where A_1, A_2 are new atomic concepts and a is new constant), and check whether a is in the answer to the query $q(x) \leftarrow A_2(x)$. Similarly, to check whether $\mathcal{K}$ implies $A \sqsubseteq C$, we can simply add the assertion $A(a)$ to

[11] `http://www.w3.org/TR/owl-features/`

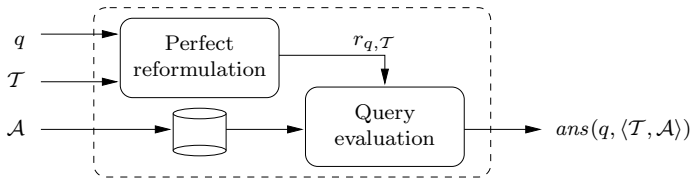

Figure 1: Query answering via query evaluation

the ABox (where a is new constant), and check whether a is in the answer to the query $q(x) \leftarrow C'(x)$, where C' is the conjunction of atoms corresponding to the concept C.

Instead, in order to exploit query evaluation for query answering, and also properly take into account that the size of the ABox (i.e., the data) largely dominates the size of the TBox, we consider the query answering process as divided in two steps (cf. Figure 1):

1. First, considering the TBox $\mathcal{T}$ only, the user query q is reformulated into a new query $r_{q,\mathcal{T}}$ (expressed in a suitable query language Ł_Q).
2. Then, the reformulated query $r_{q,\mathcal{T}}$ is evaluated over the ABox $\mathcal{A}$ only (considered as a first-order structure, i.e., a database), producing the requested answer $ans(q, \langle \mathcal{T}, \mathcal{A} \rangle)$.

As shown in [CDGL$^+$05b, CDGL$^+$05a], one of the distinguishing features of *DL-Lite* is that the above described two step query answering process makes sense, and allows us to be efficient in the size of the data. Indeed, the perfect reformulation $r_{q,\mathcal{T}}$ of a conjunctive query q over a *DL-Lite* KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ can be expressed as a union of conjunctive queries, i.e., a set of select-project-join SQL queries, and hence the query evaluation step can be performed in LogSpace in the size of the ABox $\mathcal{A}$ [AHV95]. Since the size of $r_{q,\mathcal{T}}$ does not depend on $\mathcal{A}$, the data complexity (i.e., the complexity measured as a function of the size of the ABox only) of the whole query answering algorithm is LogSpace. More precisely, the following theorem hold:

Theorem 4.1 *Answering conjunctive queries in DL-Lite is* LogSpace *in the size of the ABox (data complexity),* PTime *in the size of the TBox, and exponential in the size of the query .*

We point out that, by storing the ABox under the control of an RDBMS, which can manage effectively large numbers (i.e., millions) of objects in the knowledge base, we can completely delegate the query evaluation step to an SQL engine of a Relational Data Management System (RDBMS), and to take advantage of well established query optimization strategies. Indeed, on the one hand, the ABox $\mathcal{A}$ can be managed in secondary storage by an RDBMS, since it is possible to construct a relational database $DB(\mathcal{A})$ that faithfully represents it, and on the other hand, the perfect reformulation of a select-project-join SQL query can be expressed as a set of select-project-join SQL queries that can be directly evaluated over $DB(\mathcal{A})$.

We finally observe that *DL-Lite* can capture (the DL-subset of) RDFS[12], except for role hierarchies. In fact, the query answering technique for *DL-Lite* works also for full

[12] `http://www.w3.org/TR/rdf-schema/`

Name	Syntax	Semantics
top	$\top$	$\Delta^{\mathcal{I}}$
bottom	$\bot$	$\emptyset$
nominal	$\{a\}$	$\{a^{\mathcal{I}}\}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restriction	$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
concrete domain	$p(f_1,\ldots,f_k)$ for $p \in \mathcal{P}^{\mathcal{D}_j}$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y_1,\ldots,y_k \in \Delta^{\mathcal{D}_j} :$ $f_i^{\mathcal{I}}(x) = y_i$ for $1 \leq i \leq k \ \wedge (y_1,\ldots,y_k) \in p^{\mathcal{D}_j}\}$
GCI	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
RI	$r_1 \circ \cdots \circ r_k \sqsubseteq r$	$r_1^{\mathcal{I}} \circ \cdots \circ r_k^{\mathcal{I}} \subseteq r^{\mathcal{I}}$
concept assertion	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
role assertion	$r(a,b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$

Table 1: Syntax and semantics of $\mathcal{EL}^{++}$.

RDFS extended with participation constraints (i.e., inclusion assertions with $\exists R$ on the right-hand side), and one can show that in this case query answering is indeed LogSpace. However, if we further extend RDFS with functionality assertions, it can be shown that query answering becomes NLogSpace-hard.

4.2 The Description Logic $\mathcal{EL}^{++}$

The tractability results for $\mathcal{EL}$ together with the bio-medical applications mentioned above have motivated the research on extensions of $\mathcal{EL}$: the leitmotif for this research was to extend $\mathcal{EL}$ as far as possible by adding standard DL constructors available in ontology languages like OWL, while still retaining PTime reasoning in the presence of GCIs. This has resulted in the tractable DL $\mathcal{EL}^{++}$ [BBL05], which includes transitive roles, so-called right-identities [Spa00] on roles, nominals (and thus ABoxes), and disjointness constraints on concepts.

The logic

Like other DLs, *concept descriptions* are inductively defined with the help of a set of *constructors*, starting with a set $\mathsf{N_C}$ of *concept names*, a set $\mathsf{N_R}$ of *role names*, and (possibly) a set $\mathsf{N_I}$ of *individual names*. In this section, we introduce the extension $\mathcal{EL}^{++}$ of $\mathcal{EL}$, whose concept descriptions are formed using the constructors shown in the upper part of Table 1. There and in general, we use a and b to denote individual names, r and s to denote role names, and C, D to denote concept descriptions.

The concrete domain constructor provides an interface to so-called concrete domains, which permits reference to, e.g., strings and integers. Formally, a *concrete domain* $\mathcal{D}$ is a

pair $(\Delta^{\mathcal{D}}, \mathcal{P}^{\mathcal{D}})$ with $\Delta^{\mathcal{D}}$ a set and $\mathcal{P}^{\mathcal{D}}$ a set of *predicate names*. Each $p \in \mathcal{P}$ is associated with an arity $n > 0$ and an extension $p^{\mathcal{D}} \subseteq (\Delta^{\mathcal{D}})^n$. To provide a link between the DL and the concrete domain, we introduce a set of *feature names* $\mathsf{N_F}$. In Table 1, p denotes a predicate of some concrete domain $\mathcal{D}$ and $f_1, \ldots, f_k$ are feature names. The DL $\mathcal{EL}^{++}$ may be equipped with a number of concrete domains $\mathcal{D}_1, \ldots, \mathcal{D}_n$ such that $\Delta^{\mathcal{D}_i} \cap \Delta^{\mathcal{D}_j} = \emptyset$ for $1 \leq i < j \leq n$. If we want to stress the use of particular concrete domains $\mathcal{D}_1, \ldots, \mathcal{D}_n$, we write $\mathcal{EL}^{++}(\mathcal{D}_1, \ldots, \mathcal{D}_n)$ instead of $\mathcal{EL}^{++}$.

An $\mathcal{EL}^{++}$ *constraint box (CBox)* is a finite set of *general concept inclusions (GCIs)* and *role inclusions (RIs)*, whose syntax can be found in Table 1. Note that a finite set of GCIs would commonly be called a *general TBox*. We use the term CBox due to the presence of RIs. An $\mathcal{EL}^{++}$ *assertional box (ABox)* is a finite set of *concept assertions* and *role assertions* , whose syntax can also be found in Table 1. ABoxes are used to describe a snapshot of the world.

The semantics of $\mathcal{EL}^{++}(\mathcal{D}_1, \ldots, \mathcal{D}_n)$-concept descriptions is defined in terms of an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$. The *domain* $\Delta^{\mathcal{I}}$ is a non-empty set of individuals and the *interpretation function* $\cdot^{\mathcal{I}}$ maps each concept name $A \in \mathsf{N_C}$ to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, each role name $r \in \mathsf{N_R}$ to a binary relation $r^{\mathcal{I}}$ on $\Delta^{\mathcal{I}}$, each individual name $a \in \mathsf{N_I}$ to an individual $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, and each feature name $f \in \mathsf{N_F}$ to a partial function $f^{\mathcal{I}}$ from $\Delta^{\mathcal{I}}$ to $\bigcup_{1 \leq i \leq n} \Delta^{\mathcal{D}_i}$. The extension of $\cdot^{\mathcal{I}}$ to arbitrary concept descriptions is inductively defined as shown in the third column of Table 1.

An interpretation $\mathcal{I}$ is a *model* of a CBox $\mathcal{C}$ if, for each GCI and RI in $\mathcal{C}$, the conditions given in the third column of Table 1 are satisfied. In the definition of the semantics of RIs, the symbol "$\circ$" denotes composition of binary relations. An interpretation $\mathcal{I}$ is a *model* of an ABox $\mathcal{A}$ if, for each concept assertion and role assertion in $\mathcal{A}$, the conditions given in the third column of Table 1 are satisfied.

Reasoning in the $\mathcal{EL}^{++}$

As already mentioned at the beginning of this section, the heart of motivation for $\mathcal{EL}^{++}$ is to extend the based concept language $\mathcal{EL}$ as far as possible while still retaining PTIME reasoning in the presence of GCIs. The main inference problem considered, not only in $\mathcal{EL}$ but also in DLs in general, is concept subsumption. Though other inference problems are not trivial in the presence of the bottom concept and ABoxes, it has been shown in [BBL05] that most standard inference problems are inter-reducible.

In [Baa03a], subsumption in $\mathcal{EL}$ with the presence of either acyclic or cyclic TBoxes has been shown tractable. This is achieved by characterizing the subsumption through a proper simulation on the $\mathcal{EL}$ syntax graph. Later on, Brandt has shown that tractability can still be achieved even when admitting GCIs in $\mathcal{EL}$ [Bra04]. In order to show this, a new normal form of a general TBox has been introduced. This normalization can be computed in linear time, yielding a normalized TBox of which size is linear in the size of the original one (see, e.g. [Sun05]). A saturation-based approach, based on a set of completion rules, has been proposed. This approach has recently been extended to the more expressive $\mathcal{EL}^{++}$ [BBL05]. In that paper, Baader et al. have proved that the subsumption problem remains tractable when adding the bottom concept (and thus disjointness statements), nominals (i.e., singleton concepts), a restricted form of concrete

domains (e.g., references to numbers and strings), and a restricted form of role-value maps (which can express transitivity and the right-identity rule required in medical applications [Spa00]). On the other hand, they then proved that basically, all other additions of standard DL constructors lead to intractability of the subsumption problem, and in most cases even to ExpTime-hardness.

Instance retrieval in $\mathcal{EL}^{++}$ is PTime. Conjunctive query answering has not been studied yet neither in in $\mathcal{EL}$ nor in $\mathcal{EL}^{++}$. However the results in [CDGL$^+$05a] show us that such a service is PTime-hard in data complexity and hence cannot be delegated to a relational DBMS (actually such a lower bound holds already for instance retrieval).

Discussion

To the surprise of many, $\mathcal{EL}^{++}$, which has been proposed as a lightweight ontology language, are polytime decidable. Though this description logic can be used to completely formulate real-world biomedical ontologies, such as the Gene Ontology and SNOMED, it is worthwhile to note that neither of these realistic ontologies are composed of using GCIs. For this reason, it is interesting and natural to ask the following question:

"How far can $\mathcal{EL}$ be extended if we restrict the ontology formulation to an acyclic TBox, while still retaining polytime reasoning?"

Of course, one could hope for additional expressive powers that are important and useful for formulating real-world ontologies, when the use of general TBoxes is compromised.

4.3 Description Logics Programs

Description Logic Programs (DLP) [GHVD03] stem from the analysis of intersection between the Semantic Web approaches to rules (RuleML Logic Programs) and ontologies (OWL/DAML+OIL Description Logic). The DLP approach can be seen as an approach based on a form of safe interaction between the DL-KB and the rules: in particular, a rule language is defined such that it is possible to encode a set of rules into a semantically equivalent DL-KB.

DLP provides a significant degree of expressiveness, substantially greater than the RDFSchema fragment of Description Logic. It is a subset of both OWL-DL and the Horn fragment of First Order Logic. In fact, the standard translation of DLP axioms to First Order Logic results in Horn clauses.

We notice that the connection between DLP and logic programs is mainly syntactic. Indeed, as for the semantics, logic programming engine adopts the closed world assumption, whereas DLP adopts classical DL semantics. As for constructs allowed in the language, we point out that DLP allows for the specification of TBoxes containing concept disjointness, functionality on roles and on their inverse, (limited) inclusions on concepts, transitivity of roles, and range properties.

Logical implication in DLP is ExpTime wrt the TBox, while instance retrieval and conjunctive query answering is ExpTime in the size of the TBox and PTime (and also actually PTime-hard [CDGL$^+$05a]) in the size of the ABox (data complexity).

4.4 Horn-$\mathcal{SHIQ}$

Horn-$\mathcal{SHIQ}$ [HMS05] is a fragment of the expressive DL $\mathcal{SHIQ}$, for which reasoning, i.e., instance checking, can be encoded in a suitable positive Datalog program, i.e., Horn clauses.

Essentially Horn-$\mathcal{SHIQ}$ is obtained from $\mathcal{SHIQ}$ by allowing only axioms of the form $\bigcap C_i \sqsubseteq D$ are allowed, where each C_i has the form A or $\exists R.A$, and D has the form A, $\bot$, $\exists R.A$, $\forall R.A$, $(\geq nR.A)$, $(\leq 1R)$.

Data complexity of instance retrieval in such a logic is in PTIME-complete [HMS05], while data complexity of conjunctive query answering is still open (though obviously PTIME-hard).

4.5 Discussion and related work

Languages presented in this section are different fragments of expressive DLs with assertions and inverses studied in the 90's (cf. Section 2and 3), which are at the base of current ontology languages such as OWL, and for which optimized automated reasoning systems have been developed. Indeed, one could use, off-the-shelf, a system like Racer to perform KB satisfiability, instance checking, subsumption, logical implication of inclusion assertions, in the DLs presented in this section.

Also, reasoning with conjunctive queries in these DLs has been studied (see e.g., [CDGL00]), although not yet implemented in systems. Unfortunately, the reasoning procedures for these DLs are all EXPTIME-hard, and more importantly they are not tailored towards obtaining tight complexity bounds with respect to data complexity. Conjunctive queries combined with DLs were also considered in [DLNS98], but again data complexity was not the main concern.

Alternative reasoning procedures that allow for clearly isolating data complexity have recently been proposed, but how they will work in practice still needs to be understood: in [HMS05], a coNP upper bound for data complexity of instance checking in an expressive DL has been shown, and a polynomial fragment has been isolated, though it is open whether the technique can be extended to deal efficiently with conjunctive queries; building on the technique proposed in [LR98], coNP-completeness of answering conjunctive queries for an expressive DL with assertions, inverse roles, and number restrictions (that generalize functionality) has been shown in [OCE06].

There has been a lot of work in DLs on the boundary between polynomial and exponential reasoning. Such a work first concentrated on DLs without the TBox component of the knowledge base, and led to the development of simple DLs, such as $\mathcal{ALN}$, that admit polynomial instance checking. However, for minor variants of $\mathcal{ALN}$, such as $\mathcal{ALE}$ (where qualified existential is introduced and number restrictions are dropped), $\mathcal{FLE}^-$ (where additionally negated atomic concepts are dropped), and $\mathcal{ALU}$ (where union is introduced and number restrictions are dropped), instance checking, and therefore conjunctive query answering, is coNP-complete in data complexity [DLNS94]. We notice that, the argument used in the proof of coNP-hardness of $\mathcal{ALE}$, $\mathcal{FLE}^-$, and $\mathcal{ALU}$ in [DLNS94], immediately implies that answering conjunctive queries is coNP-hard in data complexity, if we extend *DL-Lite* with one of the following features: (1) either $\forall R.A$ or $\neg A$ can appear in the

left-hand side of inclusion assertions; (2) either $\forall R.A$ or $\neg A$ can appear as atoms in the query; (3) union of concepts can appear in the right-hand side of inclusion assertions. If we allow for cyclic inclusion assertions in the TBox, then even subsumption in CLASSIC and $\mathcal{ALN}$ (and in fact $\mathcal{FL}_0$) becomes intractable.

5 Rules in Ontologies

In this section we review the work on rules in ontology languages and in particular in description logics. On of the main motivation for looking at rules comes from the Semantic Web. The need for integrating rules within the Semantic Web framework was clear since the early developments. However, up to the last few years, the research community focused its efforts on the design of the so called *Ontology Layer*. Nowadays, this layer is fairly mature in the form of Description Logics based languages such as OWL-Lite and OWL-DL, which are now among W3C recommendations.

One of the key features of Semantic Web ontology languages development is the attention to the computational properties of the main reasoning tasks. In particular, decidability is seen as one of the characteristics which should be preserved by these languages. This constraint led to the restriction of the expressivity of ontology language which can be heavy for certain applications (e.g. Web Services, or integration of information systems). The problem increasing the expressivity of Semantic Web ontology languages over the established Ontology Layer, together with the need of providing powerful query languages, directed the research towards the investigation of the possibility of combining OWL languages with Rules based languages.

In recent years, more research has been devoted towards the integration of different sorts of rule based languages on top of the ontology layer provided by the OWL languages and in more general terms on top of a generic DL, and this work already produced some proposals for extending OWL languages. However, these proposals comes from different research communities, and often are difficult to compare because of the diverse underlying semantic assumptions.

We reckon that, since the early 90s, the Description Logics community produced several important results w.r.t. the problem of integrating DL languages and rules [DLN$^+$98, DNR97, DLN$^+$98, DLNS98, DNR02, Ros98, LR98]. For this reason we do not restrict our analysis to proposals in the context of Semantic Web. On the contrary, we show that a careful analysis of this body of work provides a valuable reference to explore the borders of expressivity and tractability of the combination of the two kinds of language.

Generally speaking we can identify three different approaches, when dealing with rules: the axiom-based approach, the logic programming approach, and the autoepistemic approach. In this section we provide a characterization of the three approaches, together with a correspondence among relevant fragments in the three cases.

5.1 Rule-extended knowledge bases

Let us consider a first-order function-free language with signature $\mathcal{A}$, and a description logic (DL) knowledge base Σ with signature subset of $\mathcal{A}$.

We do not introduce any particular DL formalism. In our context, DL individuals correspond to constant symbols, DL atomic concepts and roles (and features) are unary and binary predicates in the case of a classical DL or a OWL language, and DL atomic n-ary relations correspond to predicates of arity n in the case of a $\mathcal{DLR}$-like DL. Note that description logics with concrete data-types (such as OWL-Lite) are allowed as well.

A *term* is any constant in $\mathcal{A}$ or a variable symbol. If R is a predicate symbol of arity n and $t_1, \ldots, t_n$ are terms, $R(t_1, \ldots, t_n)$ is an *atom*, and an atom $R(t_1, \ldots, t_n)$ or a negated atom $\neg R(t_1, \ldots, t_n)$ are *literals*. A *ground* literal is a literal involving only constant terms. A set of ground literals is *consistent* if it does not contain an atom and its negation. If l is a literal, l or *not* l are *NAF-literals* (negation as failure literals). DL atoms, DL literals, and DL NAF-literals are atoms, literals, and NAF-literals whose predicates belong to the DL signature. A *rule* r may be of the forms:

$$
\begin{array}{lcll}
h_1 \wedge \ldots \wedge h_\ell & \leftarrow & b_1 \wedge \ldots \wedge b_m & \text{(classical rule)} \\
h_1 \vee \ldots \vee h_\ell & :- & b_1 \wedge \ldots \wedge b_m \wedge \mathit{not}\ b_{m+1} \wedge \ldots \wedge \mathit{not}\ b_n & \text{(lp-rule)} \\
h_1 \wedge \ldots \wedge h_\ell & \Leftarrow & b_1 \wedge \ldots \wedge b_m & \text{(autoepistemic rule)}
\end{array}
$$

where $h_1, \ldots, h_\ell, b_1, \ldots, b_n$ are literals. Given a rule r, we denote by $H(r)$ the set $\{h_1, \ldots, h_\ell\}$ of *head* literals, by $B(r)$ the set of *body* literals $\{b_1, \ldots, b_n\}$, by $B^+(r)$ the set of *NAF-free* body literals $\{b_1, \ldots, b_m\}$, and by $B^-(r)$ the set of *NAF-negated* body literals $\{b_{m+1}, \ldots, b_n\}$. We denote by $\mathsf{vars}(\{l_1, \ldots, l_n\})$ the set of variables appearing in the literals $\{l_1, \ldots, l_n\}$. The *distinguished variables* of a rule r are the variables that appears both in the head and in the body of the rule, i.e., $D(r) = \mathsf{vars}(H(r)) \cap \mathsf{vars}(B(r))$. A *ground rule* is a rule involving only ground literals. A rule is *safe* if all the variables in the head of the rule are distinguished. A *DL rule* is a rule with only DL literals. A *DL-safe rule* is a rule in which each variable appears in some non DL atom (i.e., positive literal) in the body. A set of literals is *tree-shaped* if its co-reference graph is acyclic; a co-reference graph includes literals and variables as nodes, and labelled edges indicate the positional presence of a variable in a literal. An *atomic* rule is a rule having a single literal in the head; a *positive-atomic* rule is a rule having a single atom in the head. A set of rules is *acyclic* if they are atomic and no head literal transitively depends on itself; a head literal h directly depends on a literal l if there is an atomic rule r with head h and with l part of the body $B(r)$. A set of rules is a *view* set of rules if each rule is atomic and no head literal belongs to the DL signature. A *rule-extended knowledge base* $\langle \Sigma, \mathcal{R} \rangle$ consists of a DL knowledge base Σ and a finite set $\mathcal{R}$ of rules.

5.2 The axiom-based approach

Let us consider a rule-extended knowledge base $\langle \Sigma, \mathcal{R} \rangle$ restricted to only classical rules.

Let I_Σ be a model of the description logics knowledge base Σ, i.e. $I_\Sigma \models \Sigma$. I is a model of $\langle \Sigma, \mathcal{R} \rangle$, written $I \models \langle \Sigma, \mathcal{R} \rangle$, if and only if I extends I_Σ with the interpretation of the non-DL predicates, and for each rule $r \in \mathcal{R}$ then

$$I \models \forall \mathbf{x}, \mathbf{y}. \exists \mathbf{z}. \left(\bigwedge B(r) \rightarrow \bigwedge H(r) \right)$$

where $\mathbf{x}$ are the distinguished variables of the rule $D(r)$, $\mathbf{y}$ are the non distinguished variables of the body ($\mathsf{vars}(B(r)) \setminus D(r)$), and $\mathbf{z}$ are the non distinguished variables of the head ($\mathsf{vars}(H(r)) \setminus D(r)$).

Let us define now the notion of logical implication of a ground literal l given a rule extended knowledge base: $\langle \Sigma, \mathcal{R} \rangle \models l$ if and only if $I \models l$ whenever $I \models \langle \Sigma, \mathcal{R} \rangle$. Note that the problems of DL concept subsumption and DL instance checking, and the problem of predicate inclusion (also called *query containment*) are all reducible to the problem of logical implication of a ground literal. Logical implication in this framework is undecidable, as it generalizes the so-called *recursive CARIN* as presented in [LR98]. Logical implication in an axiom-based rule extended knowledge base remains undecidable even in the case of atomic negation-free safe DL rules with a DL having just the universal role constructor $\forall R.\, C$. Note that logical implication in an axiom-based rule extended knowledge base even with an empty TBox in Σ is undecidable (see, e.g., [BM02]).

In order to recover decidability, we reduce the expressivity of the approach in several ways; all the following restrictions disallow non DL predicates in the rules.

Theorem 5.1 *1. If we restrict the axiom-based approach to have only DL rules with tree shaped heads and bodies and without negated atomic roles, the problem of logical implication in the rule extended knowledge base is* NExpTime*-complete with $\mathcal{ALCQI}$, OWL-Lite and OWL-DL as the underlying description logics knowledge base language.*

2. *If in addition to the above conditions, constants are disallowed from the rules, the problem of logical implication in the rule extended knowledge base is* ExpTime*-complete with any DL in* ExpTime *(such as $\mathcal{ALCQI}$ or OWL-Lite) as the underlying description logics knowledge base language.*

3. *[LR98]: If we restrict the axiom-based approach to have only acyclic atomic negation-free safe DL rules with the $\mathcal{ALCNR}$ DL as the underlying description logics knowledge base language, the problem of logical implication is decidable in* NExpTime*.*

4. *[MSS05, Ros05]: If we restrict the axiom-based approach to have only DL-safe atomic rules with any decidable DL, the problem of logical implication in the rule extended knowledge base is decidable.*

The SWRL proposal [HPS04b] can be considered as a special case of the axiom-based approach presented above. SWRL uses OWL-DL or OWL-Lite as the underlying description logics knowledge base language (which admits data types), but it restricts the rule language to safe rules and without negated atomic roles. From the point of view of the syntax, SWRL rules are an extension of the abstract syntax for OWL DL and OWL Lite; SWRL rules are given an XML syntax based on the OWL XML presentation syntax; and a mapping from SWRL rules to RDF graphs is given based on the OWL

RDF/XML exchange syntax. Logical implication in SWRL is still undecidable. The complexity results listed in Theorem 5.1 are applicable to SWRL as well.

Another way to make the axiom-based approach decidable is to reduce the expressivity of the DL, in order to disallow universal-like statements, while keeping rules cyclic.

In [LR98] it is shown that logical implication is decidable with atomic negation-free safe DL rules with the simple DL containing conjunction, disjunction, qualified existential, least cardinality and primitive negation.

In [CDGL$^+$04] a proposal is made of a very simple knowledge representation language, which captures the fundamental features of frame-based formalisms and of ontology languages for the semantic web; the precise definition of the language can be found in [CDGL$^+$04]. In this setting, it can be shown that the negation-free axiom-based approach is decidable, and the problem of logical implication of a ground literal is in ExpTime, and it is PTime in data complexity.

Conceptual graph rules [BM02] can be seen as a simple special case of an axiom-based rule extended knowledge base: CG-rules are negation-free, they do not have existential variables in the body, and Σ is TBox-free. Many decidable subclasses of CG-rules are special cases of the decidable cases presented above (but with Σ having a TBox); in particular, decidability of *range restricted CG-rules* is the TBox-free special case stated above [LR98] of atomic negation-free safe DL rules.

Finally, we mention [MSS04] as an implementation of the axiom based approach.

5.3 The Logic Programming approach

Let us consider a rule-extended knowledge base $\langle \Sigma, \mathcal{R} \rangle$ where $\mathcal{R}$ is restricted to be set of lp-rules $\mathcal{P}$ (called *program*). Two different approaches have been presented in the literature: *DL-log* [Ros05, Ros06] and *DL-programs* [ELST04]. Informally speaking, the former defines the models of the rule-extended knowledge base $\langle \Sigma, \mathcal{R} \rangle$ as being the models of both the program $\mathcal{R}$ and the DL knowledge base Σ, while the latter defines the models of the rule-extended knowledge base $\langle \Sigma, \mathcal{R} \rangle$ as being the models of the program $\mathcal{R}$ such that each DL literal is also a logical consequence of the DL knowledge base Σ. The DL-Log approach was first introduced with AL-Log [DLNS98] which is in fact a restricted case of [Ros05].

In the following we provide the semantics for two approaches as two separate definitions. The DL-log approach is restricted to the case in which rules are safe and contain only atoms instead of literals (e.g. no "classical" negation, in the LP sense). Moreover, NAF-atoms must be non-DL.

Definition 1 *(DL-log logical implication [Ros05, Ros06])*
We denote with $\mathsf{ground}(\mathcal{P})$ *the set of rules corresponding to the grounding of* $\mathcal{P}$ *with the constant symbols from* $\mathcal{A}$*. Let* I *be an interpretation for the signature* $\mathcal{A}$*.*

The projection of $\mathsf{ground}(\mathcal{P})$ w.r.t. I *is the ground program* $\Pi_I(\mathsf{ground}(\mathcal{P}))$ *defined as follows. For each rule* r *in* $\mathsf{ground}(\mathcal{P})$*:*

- *delete* r *if there is a DL atom* $b \in H(r)$ *s.t.* $I \models b$ *or there is a DL atom* $b \in B(r)$ *s.t.* $I \not\models b$*;*

- *delete each DL atom b in r s.t. $b \in H(r)$ and $I \not\models b$, or $b \in B(r)$ and $I \models b$.*

I is a model *for $\langle \Sigma, \mathcal{R} \rangle$ (written $I \models \langle \Sigma, \mathcal{R} \rangle$) iff*

- *I satisfies Σ, and*
- *the projection of I w.r.t. the non-DL signature of $\mathcal{A}$ is a stable model*[13] *of $\Pi_I(\mathsf{ground}(\mathcal{P}))$.*

A ground literal is logically implied by a rule extended knowledge base – written as $\langle \Sigma, \mathcal{R} \rangle \models l$ – if and only if whenever $I \models \langle \Sigma, \mathcal{R} \rangle$ then $I \models l$.

In the case of DL-programs rules should be atomic. However, the authors claim that the approach can be easily extended to general rules.

Definition 2 *(DL-programs logical implication [ELST04])*
The Herbrand base *of the program $\mathcal{P}$– written as $\mathcal{HB}_\mathcal{P}$ – is the set of the groundings of all the literals in $\mathcal{P}$ with all the constant symbols from $\mathcal{A}$. An* interpretation *I wrt $\mathcal{P}$ is a consistent subset of $\mathcal{HB}_\mathcal{P}$. We say I is a* model *of a ground literal l wrt the knowledge base Σ – written as $I \models_\Sigma l$ – if and only if $l \in I$ and*

$$\Sigma \models l \text{ if } l \text{ is a DL literal.}$$

We say that I is a model of a ground rule r – written as $I \models_\Sigma r$ – if and only if

- *$I \models_\Sigma H(r)$ whenever $I \models_\Sigma b$ for all $b \in B^+(r)$, and*
- *$I \not\models_\Sigma b$ for all $b \in B^-(r)$.*

I is a model of a rule-extended knowledge base $\langle \Sigma, \mathcal{P} \rangle$ – written as $I \models \langle \Sigma, \mathcal{P} \rangle$ – if and only if $I \models_\Sigma r$ for all rules $r \in \mathsf{ground}(\mathcal{P})$, and I is a stable model *of $\mathcal{P} \cup \{l \mid l \in I,\ l \text{ is DL literal}\}$.*
A ground literal is logically implied by a rule extended knowledge base – written as $\langle \Sigma, \mathcal{P} \rangle \models l$ – if and only if whenever $I \models \langle \Sigma, \mathcal{P} \rangle$ then $I \models_\Sigma l$.

The consequence of the semantics as defined in [ELST04] is that in the case of a NAF-free program, as well in the case of a program with stratified NAF, there is a unique stable model which coincides with the (canonical) minimal model. Therefore, logical implication can be reduced to model checking in this minimal model; this can be shown by adapting standard results from Datalog. So, if $I_m^\mathcal{P}$ is the minimal model of a NAF-free or stratified program $\mathcal{P}$, then $\langle \Sigma, \mathcal{P} \rangle \models l$ if and only if $I_m^\mathcal{P} \models_\Sigma l$. This does not hold for [Ros05].

In the case of the semantics as defined in [ELST04], it is possible to prove that the problem of logical implication of a DL literal in a general rule extended knowledge base is independent on the presence of the program P. This means that the DL knowledge base is unaffected by the rule system, which can be seen as built on top of the DL knowledge base. Note that this is not true for [Ros05]. On the other hand, the semantics as defined in [ELST04] rules out the possibility of reasoning by cases intertwined between the program $\mathcal{P}$ and the knowledge base Σ, as it will be exemplified in Subsection 5.5.

[13] See [GL91]

Theorem 5.2 *1. For rule extended knowledge bases as defined in [ELST04]: The combined complexity of logical implication in a rule extended knowledge base with an* ExpTime*-complete description logic (like, e.g., $\mathcal{ALCQI}$ or OWL-lite) is* ExpTime*-complete in the case of NAF-free or stratified programs and it is* NExpTime*-complete in the unrestricted case. In a rule extended knowledge base with a* NExpTime*-complete description logic (like, e.g., $\mathcal{ALCQIO}$ or OWL-DL) the complexity is* NExpTime*-complete in the case of NAF-free programs and it is* NExpTime ^{NP}*-complete in the case of stratified programs and in the unrestricted case as well. Note that [ELST04] restricts the rule syntax to have only a non-DL literal in the head.*

2. For rule extended knowledge bases as defined in [Ros05, Ros06]: logical implication in a rule extended knowledge base is decidable if the satisfiability problem is decidable in the description logic. The combined complexity of logical implication in a rule extended knowledge base with an OWL-DL knowledge base is NExpTime ^{NP}*-complete; in the case of (classical) negation free program $\mathcal{P}$ the complexity is* NExpTime*-complete.*[14]

An alternative semantics for of DL-Log could be the one where the recursive program is given a fixpoint semantics, which involves all individuals in the model, not only the ones in the Herbrand universe. With this semantics, logical implication is undecidable with any DL having the ability to state at least atomic inclusion axioms between concepts [CR03]. Note that in the case of acyclic rules, the fixpoint semantics coincides with the axiom-based semantics.

It is worthwhile mentioning at the end of this subsection three additional recent works that relate DLs with lp-rules: DLP [GHVD03] and [MSH04, Swi04]. In these papers it is shown how to *encode* the reasoning problem of a DL into a pure logic programming setting, i.e., into a rule extended knowledge base with a Σ without TBox. In the case of DLP, this is accomplished by encoding a severely restricted DL into a NAF-free (classical) negation-free DL program. In the two latter approaches, the full power of disjunctive logic programming is needed to perform the encoding of quite expressive DLs, at the cost of an exponential blow-up in space of the encoding.

5.4 The autoepistemic approach

Let us consider a rule-extended knowledge base restricted to autoepistemic rules.

Let I_Σ be a model, over the non empty domain Δ, of the description logics knowledge base Σ, i.e. $I_\Sigma \models \Sigma$. Let's define a variable assignment α in the usual way as a function from variable symbols to elements of Δ. A model of $\langle \Sigma, \mathcal{R} \rangle$ is a non empty set M of interpretations I, each one extending a DL model I_Σ with some interpretation of the non-DL predicates, such that for each rule r and for each assignment α for the distinguished variables of r the following holds:

[14] In [Ros06], it is shown that by an appropriate restriction of the DL expressiveness the data complexity does not increase w.r.t. the data complexity of the Datalog program alone (see also [CDGL$^+$05b]).

$$\Big(\forall I \in M.\ I, \alpha \models \exists \mathbf{x}.\ \bigwedge B(r)\Big) \rightarrow \Big(\forall I \in M.\ I, \alpha \models \exists \mathbf{y}.\ \bigwedge H(r)\Big)$$

where $\mathbf{x}$ are the non distinguished variables of the body ($\mathsf{vars}(B(r)) \setminus D(r)$), and $\mathbf{y}$ are the non distinguished variables of the head ($\mathsf{vars}(H(r)) \setminus D(r)$).

Let us define now the notion of logical implication of a ground literal l given a rule extended knowledge base: $\langle \Sigma, \mathcal{R} \rangle \models l$ if and only if

$$\forall M.\ (M \models \langle \Sigma, \mathcal{R} \rangle) \rightarrow \forall I \in M.\ (I \models l)$$

The autoepistemic approach was thoroughly analyzed by [DNR02], with the goal of formalizing the *constraint rules* implemented in many practical DL systems. Such rules, in fact, are simple to implement since they influence the ABox reasoning, but leave the TBox reasoning unaffected. These rules are also the basis of the recent formalizations of peer-to-peer systems [FKLS03]. As shown in [FKLS03], the autoepistemic semantics as defined above is equivalent to the context-based semantics of [GS98], and to the use of the autoepistemic operator, as defined, e.g., in [Rei92]. Using the results in [Mar99, GKWZ03], we can show that logical implication is decidable in the case of a rule extended knowledge base with DL rules with tree shaped body and heads, with the $\mathcal{ALC}$ DL; the precise complexity bounds are still unknown.

5.5 Comparing the three approaches

We first show in this subsection the conditions under which the three approaches coincide. This corresponds essentially to the case of negation-free view rule-extended knowledge bases with empty TBoxes. Note that this is the case of pure Datalog without a background knowledge base, for which it is well known that the three different semantics give rise to the same answer set.

Theorem 5.3 *[Llo87, MT91]: If we restrict a rule extended knowledge base with classical rules to view negation-free DL rules with TBox-free Σ, a rule extended knowledge base with lp-rules to NAF-free negation-free DL programs with TBox-free Σ, and a rule extended knowledge base with autoepistemic rules to view negation-free DL rules with TBox-free Σ, the semantics of the rule extended knowledge base with classical rules, with lp-rules, and with with autoepistemic rules coincide, i.e., the logical implication problem is equivalent in the three approaches.*

Theorem 5.4 *[Ros05]: If we restrict a rule extended knowledge base with classical rules to negation-free DL safe rules with arbitrary Σ, a rule extended knowledge base with lp-rules to NAF-free negation-free DL programs with TBox-free Σ, the semantics of the rule extended knowledge base with classical rules, with lp-rules, and with with autoepistemic rules coincide, i.e., the logical implication problem is equivalent in the three approaches.*

The above theorem is quite strict and it fails as soon as we release some assumption. We will show this by means of few examples. Consider the following knowledge base Σ, common to all the examples:

$\texttt{is-parent} \doteq \exists\texttt{is-parent-of}$
$\texttt{my-thing} \doteq \texttt{is-parent} \sqcup \neg\texttt{is-father}$
$\texttt{is-parent-of(john, mary)}$
$\texttt{is-parent(mary)}$

where we define, using standard DL notation, a TBox with the `is-parent` concept as anybody who is parent of at least some other person, and the concept `my-thing` as the union of `is-parent` and the negation of `is-father` (this should become equivalent to the top concept as soon as `is-father` becomes a subconcept of `is-parent`); and an ABox where we declare that John is a parent of Mary, and that Mary is parent of somebody.
Consider the following query rules, showing the effect of existentially quantified individuals coming from some TBox definition:

$\texttt{Q}_{\texttt{ax}}\texttt{(x)} \leftarrow \texttt{is-parent-of(x,y)}$
$\texttt{Q}_{\texttt{lp}}\texttt{(x)} \text{ :- } \texttt{is-parent-of(x,y)}$
$\texttt{Q}_{\texttt{ae}}\texttt{(x)} \Leftarrow \texttt{is-parent-of(x,y)}$

The query $\texttt{Q}_{\texttt{ax}}\texttt{(x)}$ returns {`john, mary`}; the query $\texttt{Q}_{\texttt{lp}}\texttt{(x)}$ returns {`john`}; the query $\texttt{Q}_{\texttt{ae}}\texttt{(x)}$ returns {`john, mary`}.
Consider now the query rules, which shows the impact of negation in the rules:

$\texttt{Q}_{\texttt{ax}}\texttt{(x,y)} \leftarrow \neg\texttt{is-parent-of(x,y)}$
$\texttt{Q}_{\texttt{lp}}\texttt{(x,y)} \text{ :- } \neg\texttt{is-parent-of(x,y)}$
$\texttt{Q}_{\texttt{ae}}\texttt{(x,y)} \Leftarrow \neg\texttt{is-parent-of(x,y)}$

The query $\texttt{Q}_{\texttt{ax}}\texttt{(mary, john)}$ returns false; the query $\texttt{Q}_{\texttt{lp}}\texttt{(mary, john)}$ returns true; the query $\texttt{Q}_{\texttt{ae}}\texttt{(mary, john)}$ returns false.
Consider now the following alternative sets of rules, which show that autoepistemic rules, unlike the axiom-based ones, do not influence TBox reasoning:

$\texttt{is-parent(x)} \leftarrow \texttt{is-father(x)}$
$\texttt{Q}_{\texttt{ax}}\texttt{(x)} \leftarrow \texttt{my-thing(x)}$

$\texttt{is-parent(x)} \Leftarrow \texttt{is-father(x)}$
$\texttt{Q}_{\texttt{ae}}\texttt{(x)} \Leftarrow \texttt{my-thing(x)}$

In the first axiom-based case, the query $\texttt{Q}_{\texttt{ax}}\texttt{(paul)}$ returns true; in the second autoepistemic case the query $\texttt{Q}_{\texttt{ae}}\texttt{(paul)}$ returns false (we assume that `paul` is an individual in Σ).

6 Non-Standard Inferences

In this section, logical formalisms used for dealing with non-standard inferences are presented. In particular we focus on:

- least common subsumer,
- most specific concept,

- matching,
- minimal rewriting,
- approximation, and
- debugging and explanations.

In the following we will discuss each of these in detail by giving the syntax and semantics of the formalisms used. Also, we will present the relevant reasoning techniques and existing results in the literature. Notice that, the above non-standard inferences have been identified and motivated as relevant by the tasks for ontology design and maintenance in the TONES Deliverable D05.

6.1 Least common subsumer

Intuitively, the least common subsumer of a given collection of concept descriptions is a description that represents the properties that all the elements of the collection have in common. More formally, it is the most specific concept description that subsumes the given descriptions. What this most specific description looks like, whether it really captures the intuition of representing the properties common to the input descriptions, and whether it exists at all strongly depends on the DL under consideration.

Let $\mathcal{L}$ be a DL. A concept description E of $\mathcal{L}$ is a *least common subsumer* (lcs) of the concept descriptions $C_1, \ldots, C_n$ in $\mathcal{L}$ ($lcs_{\mathcal{L}}(C_1, \ldots, C_n)$ for short) iff it satisfies

1. $C_i \sqsubseteq E$ for all $i = 1, \ldots, n$, and
2. E is the least $\mathcal{L}$ concept description with this property, i.e., if E' is an $\mathcal{L}$ concept description satisfying $C_i \sqsubseteq E'$ for all $i = 1, \ldots, n$, then $E \sqsubseteq E'$.

As an easy consequence of this definition, the lcs is unique up to equivalence, which justifies talking about *the* lcs. In addition, the n-ary lcs as defined above can be reduced to the binary lcs (the case where $n = 2$). Indeed, it is easy to see that $lcs_{\mathcal{L}}(C_1, \ldots, C_n) \equiv lcs_{\mathcal{L}}(C_1, \ldots, lcs_{\mathcal{L}}(C_{n-1}, C_n) \cdots)$. Thus, it is enough to devise algorithms for computing the binary lcs.

It should be noted, however, that the lcs need not always exist. This can have different reasons: (a) there may not exist a concept description in $\mathcal{L}$ satisfying (i) of the definition (i.e., subsuming $C_1, \ldots, C_n$); (b) there may be several subsumption incomparable minimal concept descriptions satisfying (i) of the definition; (c) there may be an infinite chain of more and more specific descriptions satisfying (i) of the definition. Obviously, (a) cannot occur for DLs containing the top concept. It is easy to see that, for DLs allowing for conjunction of descriptions, (b) cannot occur. An example for a DL exhibiting behavior (c) can be found in [Baa03b], where the lcs is defined w.r.t. a cyclic TBox.

It is also clear that in DLs allowing for disjunction, the lcs of $C_1, \ldots, C_n$ is their disjunction $C_1 \sqcup \ldots \sqcup C_n$. In this case, the lcs is not really of interest. Instead of extracting properties common to $C_1, \ldots, C_n$, it just gives their disjunction, which does not provide us with new information. Thus, it only makes sense to look at the lcs in sub-Boolean DLs.

For DLs whose expressive power lies between $\mathcal{FL}_0$ and $\mathcal{ALN}$, one can use the characterization of subsumption via finite languages over the alphabet of the role names to compute the lcs [Küs98]. For DLs with existential restrictions, the characterization of subsumption via the existence of certain simulation relations between description trees implies that the lcs corresponds to the product of the description trees [BKM99].

6.2 Most specific concepts

For introducing the most specific concepts, we thus need to say how individuals are described. This is done via ABoxes. An *ABox assertion* is an expression $C(a)$ or $r(a, b)$, where C is a concept, r is a role name, and a, b are from a set $\mathsf{N_I}$ of *individual names*. An *ABox* is simply a finite set of ABox assertions. In the presence of ABoxes, an interpretation $\mathcal{I}$ is required to additionally map each individual name a to an element $a^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$. Then, $\mathcal{I}$ *satisfies* an assertion $C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and an assertion $r(a, b)$ iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$. It satisfies an ABox iff it satisfies every assertion in it. We write $\mathcal{A} \models C(a)$ if every model of the ABox $\mathcal{A}$ satisfies the assertion $C(a)$.

Now, we come to the formal definition of the most specific concept. Let $\mathcal{L}$ be a DL. The $\mathcal{L}$ concept description E is the *most specific concept* (msc) in $\mathcal{L}$ of the individual a in the $\mathcal{L}$ ABox $\mathcal{A}$ ($msc_{\mathcal{L}}(a)$ for short) iff

1. $\mathcal{A} \models E(a)$, and
2. E is the least concept satisfying (i), i.e., if E' is an $\mathcal{L}$ concept description satisfying $\mathcal{A} \models E'(a)$ then $E \sqsubseteq E'$.

As with the lcs, the msc is unique up to equivalence, if it exists. In contrast to the lcs, which usually exists for standard DLs, the msc does not always exist in $\mathcal{EL}$, $\mathcal{ALN}$, and $\mathcal{ALE}$. This is due to the presence of so-called role cycles in the ABox such as $\{r(a, a)\}$. To overcome this problem, one can either approximate the MSC by bounding its role depth [KM01] or introduce ontologies with fixpoint semantics [BK98]. Both approaches have not yet been studied in the presence of ontologies.

6.3 Matching

Concept patterns are concept descriptions in which *concept variables* (usually denoted by X, Y, etc.) may occur in place of concept names. The main difference between concept names and concept variables is that the latter can be replaced by concept descriptions when applying a substitution.

For example, $D := P \ \& \ X \ \& \ \forall r.(Y \ \& \ \forall r.X)$ is a concept pattern containing the concept variables X and Y. By applying the substitution $\sigma := \{X \mapsto Q, \ Y \mapsto \forall r.P\}$ to it, we obtain the concept description

$$\sigma(D) = P \ \& \ Q \ \& \ \forall r.(\forall r.P \ \& \ \forall r.Q).$$

Let $\mathcal{L}$ be a DL. An $\mathcal{L}$ *unification problem* is of the form

$$C_1 \equiv^? D_1, \ \ldots, \ C_n \equiv^? D_n,$$

where $C_1, \ldots, D_n$ are $\mathcal{L}$ concept patterns. A *unifier* of this problem is a substitution σ such that $\sigma(C_i) \equiv \sigma(D_i)$ for $i = 1, \ldots, n$.

Matching was first considered in the context of the PhD thesis [McG96b], which is concerned with explaining to an ontology designer the results that have been returned by a reasoning system which decides satisfiability and subsumption. The main idea is to introduce *concept patterns*, which extend concept terms with *concept variables*. These variables stand for concept terms and can be used in the same way as concept names. In explanation, concept patterns are used to prune away parts of large concept terms (by replacing them with variables) in order to make an explanation more digestible to the user. Formally, the *matching problem* is defined as follows: given a concept pattern C and a concept term D, find a substitution σ of the variables in C with concept terms such that $\sigma(C)$ is equivalent to D. Apart from its use in explanation, matching can be used to find redundancies in ontologies. More precisely, such redundancies can be found by identifying multiple concept descriptions that match the same concept pattern [BN97]. In a similar way, matching can help in the integration of ontologies [BK00]. In the series of papers [BKBM99, BK99, BBK01], algorithms for matching have been developed for a number of DLs. One major limitation of current results for matching is that they only concern DLs that do not involve all Boolean concept constructors. Overcoming this limitation appears to be a very hard technical problem because, if all Boolean constructors are available, deciding concept matching becomes equivalent to deciding concept unification. The latter problem, which is addressed in [BN97], is technically very challenging and, as of now, only poorly understood. Another shortcoming of existing results on matching is that they usually only concern matching of two isolated concept terms, without taking into account the definitions from an ontology.

6.4 Minimal rewriting

In [BKM00], a very general framework for rewriting in DLs is introduced, which has several interesting instances. In order to introduce this framework, we fix a set N_R of role names and a set N_P of primitive concept names. Now, let $\mathcal{L}_s$, $\mathcal{L}_d$, and $\mathcal{L}_t$ be three DLs (the source-, destination, and TBox-DL, respectively). A *rewriting problem* is given by

- an $\mathcal{L}_t$ TBox $\mathcal{T}$ containing only role names from N_R and primitive concepts from N_P; the set of defined concepts occurring in $\mathcal{T}$ is denoted by N_D;
- an $\mathcal{L}_s$ concept description C using only the names from N_R and N_P;
- a binary relation ρ between $\mathcal{L}_s$ concept descriptions and $\mathcal{L}_d$ concept descriptions.

An $\mathcal{L}_d$ *rewriting of* C *using* $\mathcal{T}$ is an $\mathcal{L}_d$ concept description E built using role names from N_R and concept names from $N_P \cup N_D$ such that $C \rho E$. Given an appropriate ordering $\preceq$ on $\mathcal{L}_d$ concept descriptions, a rewriting E is called $\preceq$-*minimal* iff there does not exist a rewriting E' such that $E' \prec E$.

This general framework for rewriting in DLs covers at least two reasoning problems of interest: the *minimal rewriting problem* and the *approximation problem*. Approximation problem will be discussed in the subsequent subsection.

Minimal rewriting is the instance of the framework where (i) all three DLs are the same language $\mathcal{L}$; (ii) the TBox $\mathcal{T}$ is acyclic; (iii) the binary relation ρ corresponds to equivalence w.r.t. the TBox; and (iv) $\mathcal{L}$ concept descriptions are ordered by size, i.e., $E \preceq E'$ iff $|E| \leq |E'|$. The size $|E|$ of a concept description E is defined to be the number of occurrences of concept and role names in E.

Intuitively, the minimal rewriting of a concept term C with respect to an ontology $\mathcal{O}$ is a concept term C' such that C and C' are equivalent w.r.t. $\mathcal{O}$ and each concept C'' that is equivalent to C' w.r.t. $\mathcal{O}$ has at least the same length as C'. By replacing complex concept terms with concept names defined in $\mathcal{O}$, it is usually possible to rewrite concepts into much smaller ones. It was shown in [BKM00] that, in practical applications, the rewritten concept terms are usually relatively short and much more easily understood by humans than before the rewriting.

6.5 Approximation

Based on the general framework introduced in the previous subsection, this is the instance of the framework where (i) $\mathcal{T}$ is empty, and thus $\mathcal{L}_t$ is irrelevant; (ii) both ρ and $\preceq$ are the subsumption relation $\sqsubseteq$. In this case, we talk about approximation rather than rewriting. Given two DLs $\mathcal{L}_s$ and $\mathcal{L}_d$, an $\mathcal{L}_d$ *approximation* of an $\mathcal{L}_s$ concept description C is thus an $\mathcal{L}_d$ concept description D such that $C \sqsubseteq D$ and D is minimal (w.r.t. subsumption) with this property.

The case where $\mathcal{L}_s = \mathcal{ALC}$ and $\mathcal{L}_d = \mathcal{ALE}$ is investigated in [BKT02b]. Recall that the only difference between $\mathcal{ALC}$ and $\mathcal{ALE}$ is that disjunction is disallowed in $\mathcal{ALE}$ concept descriptions.[15] If C_1, C_2 are $\mathcal{ALE}$ concept descriptions, then it is easy to see that the approximation of the $\mathcal{ALC}$ concept description $C_1 \sqcup C_2$ by an $\mathcal{ALE}$ concept description is $lcs_{\mathcal{ALE}}(C_1, C_2)$. This suggests the following approach for approximating an $\mathcal{ALC}$ concept description C by an $\mathcal{ALE}$ concept description: just replace every disjunction in C by an application of the lcs operation. The following example demonstrates that this approach is too naïve: let $C := (\forall r.B \sqcup (\exists r.B \sqcap \forall r.A)) \sqcap \exists r.A$. If we replace the disjunction by an lcs operation and then compute the lcs, we obtain the $\mathcal{ALE}$ concept description

$$lcs_{\mathcal{ALE}}(\forall r.B, (\exists r.B \sqcap \forall r.A)) \sqcap \exists r.A \equiv \top \sqcap \exists r.A \equiv \exists r.A.$$

However, this concept description is too general. It is easy to see that $C \sqsubseteq \exists r.(A \sqcap B) \sqsubset \exists r.A$. In fact, $\exists r.(A \sqcap B)$ is the correct approximation.

In order to overcome this problem, the $\mathcal{ALC}$ concept description has to be transformed into an appropriate normal form. Basically, this normal form is obtained by distributing conjunctions over disjunctions, and by applying the rule $\forall r.C \ \ \& \ \ \forall r.D \rightarrow \forall r.(C \ \ \& \ \ D)$. For the example from above, the normal form is

$$C \equiv (\forall r.B \ \ \& \ \ \exists r.A) \sqcup (\exists r.B \ \ \& \ \ \forall r.A \ \ \& \ \ \exists r.A),$$

and $lcs_{\mathcal{ALE}}(\forall r.B \ \ \& \ \ \exists r.A, \exists r.B \ \ \& \ \ \forall r.A \ \ \& \ \ \exists r.A) = \exists r.(A \sqcap B)$.

[15] Here we assume without loss of generality that all $\mathcal{ALC}$ concept descriptions are in negation normal form where negation occurs only in front of concept names.

However, even for $\mathcal{ALC}$ concept descriptions in this normal form, one cannot simply replace disjunction by the lcs operation to obtain their $\mathcal{ALE}$ approximation. Consider the $\mathcal{ALC}$ concept description $C' = \exists r.A \sqcap \exists r.B \sqcap \forall r.(\neg A \sqcup \neg B)$. If we simply replace the disjunction by the lcs, then we obtain $\exists r.A \sqcap \exists r.B \sqcap \forall r.\top \equiv \exists r.A \sqcap \exists r.B$. However, C' is also subsumed by the more specific $\mathcal{ALE}$ concept description $\exists r.(A \sqcap \neg B) \sqcap \exists r.(B \sqcap \neg A)$. This problem can be overcome by also propagating value restrictions onto existential restrictions. An approximation algorithms based on these ideas is described in [BKT02b]. It is shown that every $\mathcal{ALC}$ concept description has an $\mathcal{ALE}$ approximation, and this approximation is unique up to equivalence, i.e., there is always a least approximation. However, the size of the approximation may grow exponentially with the size of the input description. The algorithm for computing the approximation given in [BKT02b] runs in doubly-exponential time, and it is not clear whether this time bound can be improved. In [BKT02a], these results are extended to the approximation of $\mathcal{ALCN}$ concepts descriptions by $\mathcal{ALEN}$ concept descriptions.

6.6 Debugging and explanations

Now that OWL is a W3C Recommendation, one can expect that a much wider community of users and developers will be exposed to expressive description logics. These users and developers are likely not to have a lot of experience with knowledge representation (KR), much less logic-based KR. For such people, having excellent documentation, familiar techniques, and helpful tools is a fundamental requirement.

OWL-based systems typically offer a set of standard inference services, such as concept satisfiability, classification and query answering. These services typically aim at determining whether a certain sentence logically follows from an ontology. However, most reasoners simply report these entailments, without providing a *justification* for them. Thus, the diagnosis and resolution of potential bugs is not supported.

The Ontology Engineering community widely agrees on the importance of helping the users understand the output of a reasoner. As increasingly large number of OWL ontologies become available on the Semantic Web, finding the cause of errors becomes an extremely hard task, even for experts.

The first steps in this direction, in the context of Description Logics, were taken in the early and mid nineties by the developers of the CLASSIC system [BBMR89a]. The explanation component in CLASSIC [McG96a] generated formal proofs for an inference using a deductive framework based on 'natural semantics' style proof rules that needed to be explicitly stated for the DL.

A different view was recently explored in [SC03], where the trace of a tableau reasoner was exploited to maintain a dependency relation between axioms in the KB and the inferences drawn from it. The motivation was debugging unsatisfiable concepts in the DICE terminology and the work applied to unfoldable $\mathcal{ALC}$ TBoxes. The paper provided a a new service, called *axiom pinpointing*, which, given an unsatisfiable concept, provided the minimal sets of axioms in the ontology sufficient for its unsatisfiability.

In [KPSH05], the authors extended this technique to the more expressive Description Logic $\mathcal{SHIF}$, provided an optimized implementation in the reasoner Pellet and a UI in the ontology editor Swoop. A user study showed that axiom pinpointing is effective for

debugging unsatisfiable concepts.

Recently, the *axiom pinpointing service* has been extended in various ways:

1. to deal with arbitrary entailments , such as concept subsumption and realization. In particular, given an ontology and one of its logical consequences (say, a subsumption relation) the service exposes to the users the sets of axioms in the ontology responsible for the entailment under consideration.

2. to provide finer grained justifications. The axiom pinpointing service suffers from a fundamental *granularity* problem: since it works at the asserted axiom level, it fails to determine which *parts* of the asserted axioms are *irrelevant* for the particular entailment under consideration to hold. In [KPCG05] the authors have proposed a notion of *precise justifications*, which provide a finer-grained explanations.

7 Standard Ontology Languages: OWL 1.1

We conclude this document by reviewing the work on standard ontology languages. In particular, we focus on the recent variant of OWL, named OWL 1.1, and consider both fragments of it and possible extensions.

OWL is an ontology language, or rather a family of three ontology languages, developed by the World Wide Web Consortium (W3C) as part of its Semantic Web activity [PSHH04]. The development of OWL was motivated by the key role foreseen for ontologies in the Semantic Web (i.e., providing precisely defined and machine processable vocabularies that can be used in semantically meaningful annotations), and the recognition that existing web languages, such as RDF and RDF Schema, were not expressive enough for this task [HPSvH03]. The design of OWL was heavily influenced by research in description logics (DLs); investigations of (combinations of) DL language constructors provided a detailed understanding of the semantics and computational properties of, and reasoning techniques for, various ontology language designs [BCM$^+$03, HST99, HS01, HS05]; this understanding was used to ensure that, for two of the three OWL dialects (OWL DL and OWL Lite), key reasoning problems would be decidable. Basing the language on a suitable DL also allowed for the exploitation of existing DL implementations in order to provide reasoning services for OWL applications [Hor98b, PS98, HM01c].

The standardisation of OWL has led to the development and adaption of a wide range of tools and services. These include reasoners such as FaCT++ [TH04], Racer [HM01c] and Pellet [SPC$^+$05], and editing tools such as Protégé [GMF$^+$03], Swoop [KPS$^+$05], Ontotrack [LN04] and OilEd [BHGS01]. Editing tools typically use a reasoner to compute the class hierarchy, alert users to problems such as inconsistent classes, and answer queries; several now include sophisticated debugging tools such as explanation (of inconsistency and subsumption) [KPSH05].

Although OWL was initially designed for use in (the development of) the Semantic Web, it has rapidly become a de facto standard for ontology development in general, see, e.g., OBO (`http://obo.sourceforge.net/`) and BioPax (`http://www.biopax.org/`). This is probably due to the ready availability of a wide range of OWL tools, and the greatly increased potential for sharing and reuse provided by the adoption of a standard.

OWL ontologies are now under development and/or in use in areas as diverse as e-Science, medicine, biology, geography, astronomy, defence, and the automotive and aerospace industries. Although this represents a considerable success story for OWL, such widespread use of the language has also revealed deficiencies in the original design, and led to requirements for language extensions. These included increased expressivity with respect to properties, number restrictions, and data-values, and some form of meta modelling [Mot05].

On studying these requirements, it became clear that several of them were addressed, at least in part, by recent developments in DL languages and reasoning techniques. This led to the idea to develop an incremental extension of OWL, provisionally called OWL 1.1, that would exploit these recent developments in order to provide a more expressive language, but one which retained OWL's desirable computational properties (in particular decidability) and which would allow for the relatively easy extension of existing reasoning systems in order to provide support for the new language.

7.1 Overview

The initial design of the OWL was (understandably) quite conservative, and features that did not receive widespread support within the working group were excluded from the language. Features for which effective reasoning methods were not known (or expected to be shortly known) were also not included.

As mentioned above, the use of OWL, particularly the OWL DL species of OWL, has identified several important features, support for which would greatly increase the utility of the language. Some of these, such as *qualified number restrictions*, were already supported by DL systems when OWL was designed, but were excluded from the language. Others, such as *complex role inclusion axioms*, could now be supported (at least in part) as a result of recent advances in DL theory.

For these reasons, it was decided at the first OWL: Experiences and Directions workshop (`http://www.mindswap.org/2005/OWLWorkshop/`) to design an extension to the OWL DL species of OWL, called OWL 1.1, a simple extension to OWL DL that:

1. adds language features commonly requested by users of OWL DL;
2. is known to be decidable, and for which practical decision procedures have been designed; and
3. is likely to be implemented by the developers of OWL DL reasoners.

The user requirements that drove the extensions in OWL 1.1, and the language features that address them, fall into five distinct categories:

Syntactic sugar: Some commonly used representations are difficult and/or cumbersome to express in OWL. For example, it is very common to assert that a number of classes are pairwise disjoint (this is often the default for the direct subclasses of a common parent class). Although this can be expressed in OWL, it is necessary to assert each pairwise disjointness separately (or to employ some representational "tricks"), which is cumbersome when large numbers of classes are involved, and may also make it more difficult for reasoners to optimise the way they deal with such sets of disjoint classes.

Increased expressiveness in property constructs: Although OWL is relatively expressive, there are still many situations that are difficult or impossible to represent using OWL. In particular, while OWL provides a wide range of constructors for building complex classes, relatively little can be said about properties. A very common requirement is to express the "propagation" of one property along another property [PL94, Spa00, Rec02], e.g., it may be useful to express the fact that certain locative properties are transfered across certain part-whole properties so that, when using a medical terminology ontology, a trauma or lesion located in a part of a body structure is recognised as being located in the body structure as a whole. This enables highly desirable inferences such as a fracture of the neck of the femur being inferred to be a kind of fracture of the femur, or an ulcer located in the gastric mucosa being inferred to be a kind of stomach ulcer.

Increased datatype expressiveness: OWL provides very limited features for describing classes whose features include concrete values such as integers and strings. It is a common requirement, for example, to express value ranges (a Gale is a wind whose speed is in the range 34–40 knots), or relationships between values (a carry-on bag is one where the sum of height, width, and depth does not exceed 45 inches).

Meta-modelling: Meta-modelling, i.e., the treatment of classes, properties and other entities as individuals, is allowed in some representation languages, including the OWL Full species of OWL, but was forbidden in OWL DL because of the computational difficulties that it may cause. However, users often say that they want some aspects of meta-modelling, at least the ability to associate simple information with classes such as synonyms, names in different languages, responsible person, etc. Annotation properties were added to OWL to partly satisfy this requirement, but the limited meta-modelling facilities provided by annotation properties have not satisfied users, particularly as annotation properties cannot be range-restricted.

Semantic-free comments: Annotations provide for the ability to include what might otherwise be considered "comments", such as information about the author or version number of a class. In OWL, however, this information has semantic import, and some counter-intuitive aspects, such as a membership in a class not being inferable simply because the class has a different version number. This has lead to the desire to have true comments, i.e., information associated with classes, etc., that has no semantic import at all.

7.2 Influences

OWL 1.1 has borrowed heavily from recent research on Description Logics as well as from recent research on the nature of the Semantic Web.

7.2.1 Description Logics

OWL DL is based on a description logic called $\mathcal{SHOIN}$. Even when OWL DL was designed, there were discussions as to whether it should be based on $\mathcal{SHOIN}$ or on $\mathcal{SHOIQ}$ [HS05], the latter being the former's extension with *qualifying number restrictions*. This expressive means is rather useful for modeling [WBH+05], and is known to be no more

problematical for a reasoner than the unqualified number restrictions in OWL DL. Interestingly, an effective decision procedure for $\mathcal{SHOIN}$ and $\mathcal{SHOIQ}$ has only been designed recently [HS05], and is already implemented in reasoners for OWL DL. Additionally, there have recently been two streams of work on extensions to the Description Logic underlying OWL DL.

Firstly, there has been considerable work on how best to add datatypes and relationships between data values to OWL-like languages [Lut03, PH05a]. The general ideas and requirements are basically similar, but the various proposals differ in detail: the datatypes themselves need to be "admissible" (roughly speaking, this means that datatype predicates are closed under negation and that the satisfiability of conjunctions of these predicates is decidable), which ensures that we can use datatype solvers as blackboxes in OWL reasoners.

Secondly, extensions to expressive description logics allowing more expressive property constructs have been devised and investigated. This line of work has lead to the $\mathcal{RIQ}$ [HS03], $\mathcal{SRIQ}$ [HKS05] and $\mathcal{SROIQ}$ [HKS06] description logics, and effective reasoning processes for them.

The existence of this work in the Description Logic community has made it simple to add qualified number restrictions, enhanced property constructs, and more expressive datatypes. OWL 1.1 essentially takes this work in its entirely and without significant modification.

7.2.2 OWL

OWL 1.1 and OWL DL: As OWL 1.1 is a simple extension to OWL DL, it borrows heavily from OWL DL. To this end, OWL 1.1 uses the same basic syntax style as the "abstract" syntax for OWL DL [PSHH04]. As well as using the same syntactic style, OWL 1.1 incorporates the entire OWL DL syntax, only providing extensions to it. In this way, any legal OWL DL ontology is also a legal OWL 1.1 ontology.

As well, the meaning of OWL 1.1 is compatible with the meaning of OWL DL. Instead of providing a direct model-theoretic semantics, the meaning of OWL 1.1 is provided by a mapping to the Description Logic $\mathcal{SROIQ}$ [CG05a]. This method of providing a semantics for OWL 1.1 gives more direct access to the theoretical results concerning $\mathcal{SROIQ}$, and is foreshadowed by the work of Horrocks and Patel-Schneider reducing OWL DL entailment to Description Logic satisfiability [HPS04a].

OWL 1.1 and OWL Full: OWL 1.1 does not provide any significant features provided by OWL Full over OWL DL. This is largely because OWL 1.1 is essentially a Description Logic, and the facilities provided by OWL Full over OWL DL (meta-modelling, blending objects and datatypes, unusual syntactic forms, subverting basic constructs, etc.) are essentially those that go outside of the Description Logic paradigm.

The only significant aspect of OWL Full that shows up in OWL 1.1 is meta-modelling. However, OWL 1.1 provides meta-modelling facilities via *punning*, which is not compatible with the meta-modelling features of OWL Full (which are the same as those provided by RDF). See Section 7.2.3 for more on how meta-modelling distinguishes OWL 1.1 from RDF and OWL Full.

OWL 1.1 and OWL Lite: Expressive ontology languages, such as OWL 1.1 and OWL DL, though decidable, have a high worst-case computational complexity[16] and are hard to use and implement efficiently. The design of simpler ontology languages with more tractable inferences was considered of primary importance by the W3C Web Ontology Working Group. The OWL Lite subset of OWL DL was designed as a language that is easier to use and present to naive users, as well as easier to implement.

The Web Ontology Working Group concluded that the main complexity of OWL DL relies on the possibility of defining complex boolean descriptions using, for example, union and complement; as a consequence, OWL Lite explicitly prohibits unions and complements in the definition of concepts; additionally, OWL Lite limits all descriptions in the scope of a quantifier to concept names, does not allow individuals to show up as concepts, and limits cardinalities to 0 and 1. The goal was to significantly reduce the number of available modeling constructs, on the one hand, and to eliminate the major sources of non-determinism in reasoning, on the other hand.

Although OWL Lite looks much simpler than OWL DL, it is still possible to express more complex concept descriptions by introducing new concept names, exploiting implicit negations and using axioms to associate multiple descriptions with a given concept name. So, from a user perspective, OWL Lite is even harder to use than OWL DL, since the available modeling constructs do not correspond to the actual expressivity of the language. Also, from a computational perspective, OWL Lite is only slightly less complex than OWL DL (namely ExpTime-complete instead of NExpTime-complete [Tob01]), and all the important reasoning problems remain intractable.

In contrast to OWL, OWL 1.1 does not single out just one language subset. Instead, various subsets of OWL 1.1 have been identified, each of which benefits from tractable (i.e., polynomial time) reasoning for one or more important reasoning tasks [CG05b]. The intention is that these subsets can be used and implemented as appropriate to a particular application.

7.2.3 RDF

OWL 1.1 diverges from the same-syntax extension of RDF vision of the Semantic Web, as embodied in RDFS and OWL Full. Like all species of OWL, OWL 1.1 uses URI references for its names and thus fits well into the Semantic Web. However, OWL 1.1 is not compatible with RDF, and thus is not compatible with OWL Full. There are two areas of incompatibility.

OWL 1.1 includes semantic-free comments. In RDF, as in OWL Full, all information is in the form of triples, and all triples have semantic import. This makes it impossible to include syntactic-only comments that can survive transmission or processing.

OWL 1.1 uses a (weak) form of meta-modelling called *punning*. In punning, names can be used for several purposes; for example, *Person* can at the same time be the name of a class and the name of an individual. The different uses of a name are, however, completely independent, and from a semantic point of view they can be thought of as separate names, e.g., *Person-the-Class* and *Person-the-Individual*.

[16] Satisfiability and subsumption are NExpTime-complete for $\mathcal{SHOIQ}$, and ExpTime-complete for $\mathcal{SHIQ}$ [Tob01].

Punning is compatible with annotation properties as used in OWL DL, as annotation properties were expressly designed so that their use would not have any effect on class level entailments. However, punning is not compatible with the meta-modelling possibilites inherent in the semantics of RDF [Hay04] (and thus inherent in OWL Full), precisely because it makes the two uses of a name semantically independent.

A triple syntax is being provided for OWL 1.1, syntactically compatible with the triple syntax for OWL DL. However, for the above reasons, this syntax could not be given a meaning compatible with the RDF meaning for triples, at least not without some very difficult semantic tiptoeing (such as providing comprehension principles for comments that essentially added every possible comment to every element of the domain of discourse) as well as some questionable encoding (such as creating fresh URI references for punning purposes, e.g., using *Person-the-Class* and *Person-the-Individual* instead of just *Person*). The appropriateness of continuing along this line with OWL 1.1 is called even more into question by the impossibility of extending it to Semantic Web languages with expressive power on a par with that of First-Order Logic [PS05].

7.3 Specification

OWL 1.1 is a complete logic, and thus come with a syntax and a (model-theoretic) semantics. Actually OWL 1.1 has two different syntaxes, the one described here and an XML syntax. We only provide here the extensions to the OWL DL abstract syntax [PSHH04].

Syntax for OWL 1.1

The "abstract" syntax of of OWL 1.1 is an extension of the "abstract" syntax for OWL DL. The extensions in OWL 1.1 lift its expressive power to that of SROIQ [HKS06]. This amounts to adding qualified cardinality restrictions, as an extension to restrictions on datatype properties and object properties; local reflexivity restrictions for simple properties, as an extension to restrictions on object properties; reflexive, irreflexive, and anti-symmetric flags for simple properties, as an extension to the flags allowed on object properties; and disjointness of simple and datatype properties, and *regular* property inclusion axioms, as new axioms.

dataRestrictionComponent ::= *dataCardinality*
dataCardinality ::= minCardinality(*non-negative-integer dataRange*)
| maxCardinality(*non-negative-integer dataRange*)
| cardinality(*non-negative-integer dataRange*)
individualRestrictionComponent ::= *individualCardinality*
individualCardinality ::= minCardinality(*non-negative-integer description*)
| maxCardinality(*non-negative-integer description*)
| cardinality(*non-negative-integer description*)
individualRestrictionComponent ::= self
individualvaluedPropertyFlags ::= Reflexive | Irreflexive
| Symmetric | AntiSymmetric
axiom ::= DisjointProperties($datavaluedPropertyID^+$)

| DisjointProperties(*individualvaluedPropertyID*$^+$)
axiom ::= SubPropertyOf(propertyChain(*individualvaluedPropertyID*$^+$)
individualvaluedPropertyID)

Only simple properties (i.e., properties that are not implied by property chains, see [HKS06] for details) can: have the self restriction component; be specified as being Reflexive, Irreflexive, Symmetric, or Antisymmetric; or be used in DisjointProperties axioms for individual-valued properties.

The SubPropertyOf axioms involving individual-valued properties must be *regular*. That is, there must be a strict partial order $<$ on individual-valued properties such that for each SubPropertyOf axiom involving individual-valued properties, of the form SubPropertyOf(S R) S is the inverse of R, S is of the form propertyChain(R R), S is of the form propertyChain($S1$ ... Sn) and each $Si < R$, S is of the form propertyChain(R $S1$ ... Sn) and each $Si < R$, or S is of the form propertyChain($S1$ ... Sn R) and each $Si < R$.

The first couple of other additions in OWL 1.1 are simple syntactic sugar. To make the common construct of multiple disjoint classes easier to state, OWL 1.1 provides an axiom that directly states that a group of classes are pairwise disjoint, instead of having to use separate disjoint axioms for each pair of classes. Similarly, OWL 1.1 provides a construct that allows to state that an individual does not have a particular property value, instead of, e.g., having to state that the individual is an instance of a suitable restriction class.

axiom ::= DisjointUnion(*description*$^+$)
value ::= valueNot(*individualvaluedPropertyID individualID*)
| valueNot(*individualvaluedPropertyID individual*)
| valueNot(*datavaluedPropertyID dataLiteral*)

OWL 1.1 includes its own methods for user-defined datatypes, using a syntax similar to the one used in Protégé. The semantics for OWL 1.1 user-defined datatypes is taken from XML Schema Datatypes [BM01].

dataRange ::= datatype(*datatypeID* { *datatypeRestriction* })
datatypeRestriction ::= *datatypeFacet(dataLiteral*)
datatypeFacet ::= length | minLength | maxLength | pattern | enumeration
| maxInclusive | maxExclusive | minInclusive | minExclusive
| totalDigits | fractionDigits
axiom ::= Datatype(*datatypeID* { *annotation* } base(*datatypeID*)
{ *datatypeRestriction* })

Datatype facets should only by used where they would be allowed in XML Schema Datatypes, except that the length, minLength, maxLength, and pattern facets are not allowed for numeric types. Datatype facets have the same meaning as in XML Schema Datatypes, except that they uniformly work in the value space, never the lexical space. If a datatype facet is used in a way that has no meaning, such as (length 5$\wedge\wedge$xsd:string), then the datatype extension is empty.

OWL 1.1 allows restrictions that relate values for different data-valued properties on the same individual.

restriction ::= holds(*datatypePredicateID* { *argument* })
restriction ::= *datatypePropertyID* | *dataLiteral*
datatypePredicateID ::= equal | notEqual | lessThan | lessThanEqual
| greaterThan | greaterThanEqual

The syntax here allows an arbitrary number of arguments, but must be appropriate for the predicate, and all the current predicates only allow two arguments. All invalid combinations are unsatisfiable (i.e., they do not signal an error). The equality and order for a particular base type is taken from XML Schema Datatypes. If a base datatype does not have an order then the ordering restriction is unsatisfied.

OWL 1.1 removes the limitation imposed in OWL DL that the sets of class, individual and property names must be pairwise disjoint. The semantic change to allow this without computational consequences is to break the RDF-inspired connection between class and property extensions and the individual denotation of names. With this change, any name can be made the subject of a non-annotation property, but in this (syntactic) context the name is always (semantically) interpreted as an individual. As simple syntactic sugar, non-annotation properties can be used where annotations are allowed in OWL DL.

annotation ::= *value* | type(*description*)

A class or property axiom with an annotation is syntactic sugar for an extra Individual axiom relating the class or property name to the annotations.

OWL 1.1 allows arbitrary comments to be inserted in ontologies.

comment ::= Comment({ *dataLiteral* | *URIreference* })

A comment is allowed anywhere white space is allowed. Comments have no semantic import in OWL 1.1, but comments should survive processing and transmission by OWL 1.1 systems.

Semantics for OWL 1.1

The semantics for OWL 1.1 rely on a translation into the description logic $\mathcal{SROIQ}(\mathcal{D}^+)$, which extends the logic $\mathcal{SROIQ}$ [HKS06] with datatypes and datatype restrictions. A similar translation was used to define the semantics of Standard OIL in terms of the Description Logic $\mathcal{SHIQ}(D)$ [FvHH+01].

Since OWL 1.1 is an extension of OWL-DL (in the same way that $\mathcal{SROIQ}(\mathcal{D}^+)$ is an extension of $\mathcal{SHOIN}(\mathcal{D})$), this document also provides a well-defined semantics for OWL-DL documents that is equivalent to the direct model-theoretic semantics given in the OWL documentation [PSHH04]. Although both semantics are equivalent, a translation-based semantics has several advantages with respect to a direct semantics: a translation-based approach results in a cleaner, simpler and more precise specification; it gives direct access to theoretical results for the logic; and it provides a direct implementation pathway.

We will define a *translation* function that maps OWL 1.1 ontologies into equivalent $\mathcal{SROIQ}(\mathcal{D}^+)$ knowledge bases. The translation function and the semantics of $\mathcal{SROIQ}(\mathcal{D}^+)$ completely specify the semantics of OWL 1.1. The semantics of $\mathcal{SROIQ}$,

OWL 1.1 Abstract Syntax	$\mathcal{SROIQ}(\mathcal{D}^+)$ Syntax
A (Class URI)	Concept name A
owl:Thing	$\top$
owl:Nothing	$\bot$
R (Object Property URI)	Abstract role R
U (Datatype Prop. URI)	Concrete role U
a (Individual URI)	Individual a
ϕ (Data Value or Plain Literal)	Concrete value ϕ
p_m (Datatype Predicate ID)	Concrete Predicate p_m
Φ (OWL 1.1 Datatype)	Supported Datatype Φ
intersectionOf($C_1 ... C_n$)	$\sigma(C_1) \sqcap ... \sqcap \sigma(C_n)$
unionOf($C_1 ... C_n$)	$\sigma(C_1) \sqcup ... \sqcup \sigma(C_n)$
complementOf(C)	$\neg\sigma(C)$
oneOf(a_1 .... a_n)	$\{\sigma(a_1)\} \sqcup ... \sqcup \{\sigma(a_n)\}$
oneOf($\phi_1 ... \phi_n$)	$\{\sigma(\phi_1), ..., \sigma(\phi_n)\}$
restriction(R $X_1 ... X_n$)	σ(restriction(R X_1)) $\sqcap$... $\sqcap$ σ(restriction(R X_n))
restriction(R someValuesFrom(C))	$\exists\sigma(R).\sigma(C)$
restriction(R allValuesFrom(C))	$\forall\sigma(R).\sigma(C)$
restriction(R hasValue(a))	$\exists\sigma(R).\{\sigma(a)\}$
restriction(R self)	$\exists\sigma(R).$Self
restriction(R minCardinality(n))	$\geq n\sigma(R).\top$
restriction(R maxCardinality(n))	$\leq n\sigma(R).\top$
restriction(R Cardinality(n))	$= n\sigma(R).\top$
restriction(R minCardinality(n C))	$\geq n\sigma(R).\sigma(C)$
restriction(R maxCardinality(n C))	$\leq n\sigma(R).\sigma(C)$
restriction(R Cardinality(n C))	$= n\sigma(R).\sigma(C)$
restriction(U $X_1 ... X_n$)	σ(restriction(U X_1)) $\sqcap$... $\sqcap$ σ(restriction(U X_n))
restriction(U someValuesFrom(Φ))	$\exists\sigma(U).\sigma(\Phi)$
restriction(U allValuesFrom(Φ))	$\forall\sigma(U).\sigma(\Phi)$
restriction(U hasValue(ϕ))	$\exists\sigma(U).\{\sigma(\phi)\}$
restriction(U minCardinality(n))	$\geq n\sigma(U)$
restriction(U maxCardinality(n))	$\leq n\sigma(U)$
restriction(U Cardinality(n))	$= n\sigma(U)$
restriction(U minCardinality(n Φ))	$\geq n\sigma(U).\sigma(\Phi)$
restriction(U maxCardinality(n Φ))	$\leq n\sigma(U).\sigma(\Phi)$
restriction(U Cardinality(n Φ))	$= n\sigma(U).\sigma(\Phi)$
restriction(holds(p_m) $U_1 ... U_m$))	$\exists\sigma(U_1)...\sigma(U_m).\sigma(p_m)$

Table 2: Translation of OWL 1.1 into $\mathcal{SROIQ}(\mathcal{D}^+)$

OWL 1.1 Abstract Syntax	$\mathcal{SROIQ}(\mathcal{D}^+)$ Syntax
ObjectProperty(R super(R_1)...super(R_m)	$\bigcup_{i=1}^{m}\{\sigma(R) \sqsubseteq \sigma(R_i)\}$
domain(C_1)...domain(C_m)	$\bigcup_{i=1}^{m}\{\geq 1\sigma(R) \sqsubseteq \sigma(C_i)\}$
range(C_1)...range(C_m)	$\bigcup_{i=1}^{m}\{\top \sqsubseteq \forall\sigma(R).\sigma(C_i)\}$
inverseOf(S)	$\{\sigma(R) \equiv \sigma(S)^-\}$
[Symmetric]	$\{\sigma(R) \equiv \sigma(R)^-\}$
[Functional]	$\{\top \sqsubseteq \leq 1\sigma(R)\}$
[InverseFunctional]	$\{\top \sqsubseteq \leq 1\sigma(R)^-\}$
[Transitive]	{Trans(σ(R))}
[Reflexive]	{Ref(σ(R))}
[Irreflexive]	{Irr(σ(R))}
[AntiSymmetric])	{ASymm(σ(R))}
SubPropertyOf(R_1 R_2)	$\{\sigma(R_1) \sqsubseteq \sigma(R_2)\}$
SubPropertyOf(propertyChain($R_1 \cdots R_m$) R)	$\{\sigma(R_1)...\sigma(R_m) \sqsubseteq \sigma(R)\}$
DisjointProperties(R_1 R_2)	{Dis(σ(R$_1$), σ(R$_2$))}
EquivalentProperties($R_1 ... R_m$)	$\bigcup_{i=1}^{m-1}\{\sigma(R_i) \equiv \sigma(R_{i+1})\}$
DataProperty(U super(U_1)...super(U_m)	$\bigcup_{i=1}^{m}\{\sigma(U) \sqsubseteq \sigma(U_i)\}$
domain(C_1)...domain(C_m)	$\bigcup_{i=1}^{m}\{\geq 1\sigma(U) \sqsubseteq \sigma(C_i)\}$
range(Φ_1)...range(Φ_m)	$\bigcup_{i=1}^{m}\{\top \sqsubseteq \forall\sigma(U).\sigma(\Phi_i)\}$
[Functional])	$\{\top \sqsubseteq \leq 1\sigma(U)\}$
SubPropertyOf($U_1 U_2$)	$\{\sigma(U_1) \sqsubseteq \sigma(U_2)\}$
EquivalentProperties($U_1 ... U_m$)	$\bigcup_{i=1}^{m-1}\{\sigma(U_i) \equiv \sigma(U_{i+1})\}$
DisjointProperties(U_1 U_2)	{Dis(σ(U$_1$), σ(U$_2$))}
Class(A partial $C_1 ... C_m$)	$\{\sigma(A) \sqsubseteq \sigma(C_1) \sqcap ... \sqcap \sigma(C_m)\}$
Class(A complete $C_1 ... C_m$)	$\{\sigma(A) \equiv \sigma(C_1) \sqcup ... \sqcup \sigma(C_m)\}$
EnumeratedClass(A $a_1 ... a_m$)	$\{\sigma(A) \equiv \{\sigma(a_1)\} \sqcup ... \sqcup \{\sigma(a_m)\}\}$
EquivalentClasses($C_1 ... C_m$)	$\bigcup_{i=1}^{m-1}\{\sigma(C_i) \equiv \sigma(C_{i+1})\}$
DisjointClasses($C_1 ... C_m$)	$\bigcup_{i,j=1;i\neq j}^{m}\{\sigma(C_i) \sqsubseteq \neg\sigma(C_j)\}$
DisjointUnion(C $C_1 ... C_m$)	$\{\sigma(C) \equiv \sigma(C_1) \sqcup ... \sqcup \sigma(C_m)\} \cup \bigcup_{i,j=1;i\neq j}^{m}\{\sigma(C_i) \sqsubseteq \neg\sigma(C_j)\}$
Individual(a type(C_1)...type(C_m)	$\bigcup_{i=1}^{m}\{\sigma(C_i)(\sigma(a))\}$
value(R_1 b_1)...value(R_m b_m)	$\bigcup_{i=1}^{m}\{\sigma(R_i)(\sigma(a), \sigma(b_i))\}$
value(U_1 ϕ_1)...value(U_m ϕ_m)	$\bigcup_{i=1}^{m}\{\sigma(U_i)(\sigma(a), \sigma(\phi_i))\}$
valueNot(R_1 b_1)...valueNot(R_m b_m)	$\bigcup_{i=1}^{n}\{\neg\sigma(R_i)(\sigma(a), \sigma(b_i))\}$
valueNot(U_1 ϕ_1)...valueNot(U_m ϕ_m))	$\bigcup_{i=1}^{m}\{\neg\sigma(U_i)(\sigma(a), \sigma(\phi_i))\}$
SameIndividual($a_1 ... a_m$)	$\bigcup_{i,j=1;i\neq j}^{m}\{\sigma(a_i) \doteq \sigma(a_j)\}$
DifferentIndividuals($a_1 ... a_m$)	$\bigcup_{i,j=1;i\neq j}^{m}\{\sigma(a_i) \neq \sigma(a_j)\}$

along with a decision procedure for reasoning in the logic, are given in [HKS06]; the semantics of a suitable datatype extension are given in [PH05a].

The translation of OWL 1.1 into $\mathcal{SROIQ}(\mathcal{D}^+)$ is quite straightforward, and follows naturally from the syntax and semantics of OWL-DL and from the syntax and informal specification of OWL 1.1 given in Section 7.3. Let $\mathcal{O}_0$ be an OWL 1.1 ontology and let $\{\mathcal{O}_j\}_{1\leq j\leq m}$ be the set of ontologies imported (directly or indirectly) by $\mathcal{O}_0$; let $\{\alpha_1^j, ..., \alpha_{n_j}^j\}$ for $0 \leq j \leq m$ be the set of axioms and facts contained in $\mathcal{O}_j$. The translation into a $\mathcal{SROIQ}$ knowledge base $\mathcal{T}$ is as follows: $\mathcal{T} = \bigcup_{0\leq j\leq m} \sigma(\mathcal{O}_j)$, where $\sigma(\mathcal{O}_j) = \bigcup_{1\leq i\leq n_j} \sigma(\alpha_i^j)$. The syntactic correspondence between OWL 1.1 descriptions and $\mathcal{SROIQ}(\mathcal{D}^+)$ concepts is given in Table 2 as is the correspondence between OWL 1.1 axioms and facts and $\mathcal{SROIQ}(\mathcal{D}^+)$ axioms. This table completely specifies the translation function σ and should be read as follows: given a construct in OWL 1.1 abstract syntax in the first column, its evaluation under σ is given in the second column.

7.4 Implementation

OWL 1.1 has been developed outside any formal standardization process. Instead, the intent was to advance the state of the deployed and used art before moving to a standards body. Several of the OWL 1.1 extensions were selected because they were already supported by some OWL tools and were deployed (or would quickly be deployed) by key users. For example, qualified number restrictions are supported by Racer, KAON2, and FaCT++ as well as the Protégé editor. Unfortunately, this support is primarily through the DIG interface and obsolete exchange formats.[17] Similarly, Pellet will reason with user defined datatypes, but it will not consume the format Protégé emits. Both these features are strongly in demand from the user community[WBH+05], but they are not used due to the lack of interoperability. Since this interoperability was mostly a matter of agreeing on a common syntax, it is likely that these features will be widely available after OWL 1.1 is finalized.

One radical if "surface" difference in OWL 1.1 is the change in syntax. In OWL 1.1, there is a normative XML syntax that is described by an XML schema. The WebOnt working group did produce a document describing a direct XML syntax for OWL, but it is incomplete and was never significantly used. We expect that the availability of a sensible XML schema friendly format will make it possible to build useful OWL 1.1 tools based on the XML infrastructure. For example, schema aware editors could be fruitfully used to edit OWL 1.1, and XPath and XSLT could be used for a variety of tasks. There is also an RDF encoding of OWL 1.1 (thus, an RDF/XML exchange format for it), so users can adopt the format that best suits their needs. Some future extensions, however, may build on the XML format (see section 7.5).

Several categories of OWL 1.1 features (syntactic sugar, semantic-free comments, meta-modelling by punning) are essentially trivial to implement, since they can be handled with a transformation into the core formalism. From an implementation perspective, the most substantial extensions in OWL 1.1 are the property constructors. In particular, the known decision procedure for $\mathcal{SROIQ}$ involves the use of automata to manage

[17] See `http://www.w3.org/2001/sw/BestPractices/OEP/QCR/` for a discussion.

the property chains. While the automata seem modular, there is only very limited experience with implementing and optimizing algorithms incorporating them [HS03]. The other extensions with regard to properties (e.g., disjoint roles or negated property assertions), while conceptually simpler (e.g., negated property roles may be encoded using nominals), also lack implementation experience. The consensus of implementors at the OWLED workshop was that these features were reasonable to implement, but the first implementations are not yet available.

Significantly, at the OWLED workshop, the major OWL reasoner implementors (those of Cerebra, RACER, FaCT++, KAON2, and Pellet) and editor implementors (Protégé and Swoop) pledged to support OWL 1.1 in a timely manner (in particular, for preliminary implementation within six months of reasonably firm specifications), and implementation work is already underway.

7.5 Future Extensions

OWL 1.1 was from the start intended to be an easy, incremental improvement to OWL. With a year and a half of experience with OWL, there was strong consensus as to the several of the obvious holes in the language. However, OWL 1.1 was also intended to start movement toward a larger extension of OWL, which, for the purposes of this paper, we shall refer to as OWL 2.0. There is a wide variety of academic and industrial research concerning expressive extensions to OWL, much of it driven by user demand, some of it driven by standardization in related areas. While it is difficult to predict what a future working group might find compelling, there are five obvious features which would be sensible to consider for the next version of OWL. **Syntactic extensibility:** Since the

OWLED workshop, there have been a number of additional proposals for syntactic sugar even beyond what OWL 1.1 offers. This suggests that some form of macro system would be useful. An obvious proposal is to center the system on the new XML syntax and make use of the extensive transformation infrastructure for XML (e.g., XPath and XSLT). Such a proposal is likely to emerge from the next OWL workshop.

Query: There are efficient implementations of some form of conjunctive ABox querying in Racer, Pellet, and KAON2. While the Data Access Working Group only defined the semantics of SPARQL queries for RDF graphs [PS06], there is a hook allowing one to plug in other semantics, for example, that of OWL. It would be straightforward to support such in OWL 2.0.

Integration with rules: Integration of rules of various sorts and DL-based ontology languages is not only a hot research area, but also a requirement for the new Rules Interchange Format (RIF) Working Group. The OWL community could define some extensions to RIF specifically designed around OWL, e.g., based on SWRL [HPS04b] or on decidable variations of SWRL [MSS04, Ros05].

Non-monotonicity: A common request for OWL is non-montonic constructs. Unfortunately, in spite of the intense interest, there is little settled consensus or practical experience with non-monontonic features in description logic systems. It may be that in the coming year the picture will become clearer, but there needs to be a more effective

gathering of grounded use cases for non-monotonicity in OWL so that the appropriate design decisions can be made.

Meta-modelling: OWL 1.1 meta-modelling does not facilitate domain modelling [Mot05], nor does it cover some useful sorts of annotative behavior [GW04]. Although the meta-modelling facilities of OWL Full were strongly argued for within the WebOnt working group, actual use of *those* particular facilities is rare [Wan06]. So, more work must be done to determine what additional meta-modelling capabilities are both feasible and will be actually used. Clearly the first three classes of feature are more ripe for standardisation than the last two. Of course, there are many other features one might hope for in the next version of OWL, for example, even more expressive datatypes, role constructors, fixed point operators and hybrid logic constructs.

7.6 Discussion

We have presented OWL 1.1, an incremental extension to OWL DL that exploits recent developments in DL languages and reasoning techniques in order to satisfy some common requirements expressed by users of OWL DL. Although some of these extensions are no more than syntactic sugar, others add real expressive power to the language, and in particular significantly extend what can be said about properties. In spite of this increased expressive power, OWL 1.1 retains the desirable computational properties of OWL DL: key reasoning problems are decidable, and practical decision procedures are available for them. Support from the implementors of prominent OWL DL reasoning and editing tools means that OWL 1.1 compatible systems should be available in the near future.

Although OWL 1.1 is backwards compatible with OWL DL, it departs significantly from OWL DL in some important respects: the semantics of OWL 1.1 is not given directly, but via a mapping to $\mathcal{SROIQ}(\mathcal{D}^+)$; OWL 1.1 uses XML Schema for its normative exchange syntax; the RDF syntax of OWL 1.1 does not fully respect the semantics of RDF, and so is not semantically compatible with OWL Full.

It is anticipated that OWL 1.1 will be only a first step, and that larger extensions will follow. These could include support for, e.g., (some form of) rules, macro and query languages. It is also anticipated that, given sufficient support from implementors and users, it may be appropriate to initiate standardisation activities for OWL 1.1 and/or OWL 2.0.

References

[AHLM91] E. Achilles, B. Hollunder, A. Laux, and J. P. Mohren. $\mathcal{KRIS}$: Knowledge Representation and Inference System – User guide. Technical Report D91-14, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), 1991.

[AHV95] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison Wesley Publ. Co., 1995.

[AvBN96] Hajnal Andréka, Johan van Benthem, and Istvaán Németi. Modal languages and bounded fragments of predicate logic. Technical Report ML-96-03, ILLC, University of Amsterdam, 1996.

[Baa90] Franz Baader. Terminological cycles in KL-ONE-based knowledge representation languages. In *Proc. of the 8th Nat. Conf. on Artificial Intelligence (AAAI'90)*, pages 621–626, Boston (Ma, USA), 1990.

[Baa91] Franz 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.

[Baa03a] F. Baader. Terminological cycles in a description logic with existential restrictions. In Georg Gottlob and Toby Walsh, editors, *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 325–330. Morgan Kaufmann, 2003.

[Baa03b] Franz Baader. Computing the least common subsumer in the description logic $\mathcal{EL}$ w.r.t. terminological cycles with descriptive semantics. In *Proceedings of the 11th International Conference on Conceptual Structures, ICCS 2003*, volume 2746 of *Lecture Notes in Artificial Intelligence*, pages 117–130. Springer-Verlag, 2003.

[BAHP82] Mordechai Ben-Ari, Joseph Y. Halpern, and Amir Pnueli. Deterministic propositional dynamic logic: Finite models, complexity, and completeness. *J. of Computer and System Sciences*, 25:402–417, 1982.

[BBH96] Franz Baader, Martin Buchheit, and Bernhard Hollunder. Cardinality restrictions on concepts. *Artificial Intelligence*, 88(1–2):195–213, 1996.

[BBK01] Franz Baader, Sebastian Brandt, and Ralf Küsters. Matching under side conditions in description logics. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 213–218, 2001.

[BBL05] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, pages 364–369, 2005.

[BBMR89a] A. Borgida, R. Brachman, D. McGuinness, and L. Resnick. Classic: A structural data model for objects. In *SIGMOD*, 1989.

[BBMR89b] Alexander Borgida, Ronald J. Brachman, Deborah L. McGuinness, and Lori Alperin Resnick. CLASSIC: A structural data model for objects. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 59–67, 1989.

[BCG01] Daniela Berardi, Diego Calvanese, and Giuseppe De Giacomo. Reasoning on uml class diagram using description logic based system. In *Proc. of the KI-2001 Workshop W6 on Applications of Description Logics ADL-01*, volume 44. http://ceur-ws.org/Vol-44/, 2001.

[BCM+03] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.

[BDNS98] Martin Buchheit, Francesco M. Donini, Werner Nutt, and Andrea Schaerf. A refined architecture for terminological systems: Terminology = schema + views. *Artificial Intelligence*, 99(2):209–260, 1998.

[BDS93a] Martin Buchheit, Francesco M. Donini, and Andrea Schaerf. Decidable reasoning in terminological knowledge representation systems. *J. of Artificial Intelligence Research*, 1:109–138, 1993.

[BDS93b] Martin Buchheit, Francesco M. Donini, and Andrea Schaerf. Decidable reasoning in terminological knowledge representation systems. In *Proc. of the 13th Int. Joint Conf. on Artificial Intelligence (IJCAI'93)*, pages 704–709, 1993.

[Ber66] R. Berger. The undecidability of the dominoe problem. *Mem. Amer. Math. Soc.*, 66:1–72, 1966.

[BFH+94] Franz Baader, Enrico Franconi, Bernhard Hollunder, Bernhard Nebel, and Hans-Jürgen Profitlich. An empirical analysis of optimization techniques for terminological representation systems or: Making KRIS get a move on. *Applied Artificial Intelligence. Special Issue on Knowledge Base Management*, 4:109–132, 1994.

[BFT95] 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.

[BH91a] Franz Baader and Philipp 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, 1991.

[BH91b] Franz Baader and Bernhard Hollunder. $\mathcal{KRIS}$: $\mathcal{K}$nowledge $\mathcal{R}$epresentation and $\mathcal{I}$nference $\mathcal{S}$ystem. *SIGART Bull.*, 2(3):8–14, 1991.

[BH92] Franz Baader and Philipp Hanschke. Extensions of concept languages for a mechanical engineering application. In *Proc. of the 16th German Workshop on Artificial Intelligence (GWAI'92)*, volume 671 of *Lecture Notes in Computer Science*, pages 132–143. Springer, 1992.

[BHGS01] Sean Bechhofer, Ian Horrocks, Carole Goble, and Robert Stevens. OilEd: A Reason-able ontology editor for the semantic web. In *Proc. of the Joint German/Austrian Conf. on Artificial Intelligence (KI 2001)*, number 2174 in Lecture Notes in Artificial Intelligence, pages 396–408. Springer, 2001. Appeared also in Proc. of the 2001 Description Logic Workshop (DL 2001).

[BHMC03] Sean Bechhofer, Volker Haarslev, Ralf Möller, and Peter Crowthe. The DIG description logic interface. In *Proceedings of the International Workshop on Description Logics (DL-2003), Rome, Italy, September 5-7*, 2003.

[BHN+92] Franz Baader, Bernhard Hollunder, Bernhard Nebel, Hans-Jürgen Profitlich, and Enrico Franconi. An empirical analysis of optimization techniques for terminological representation systems. In *Proc. of the 3rd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'92)*, pages 270–281. Morgan Kaufmann, 1992.

[BK89] François Bancilhon and Setrag Khoshafian. A calculus for complex objects. *J. of Computer and System Sciences*, 38(2):326–340, 1989.

[BK98] Franz Baader and Ralf Küsters. Computing the least common subsumer and the most specific concept in the presence of cyclic $\mathcal{ALN}$-concept descriptions. In *Proc. of the 22nd German Annual Conf. on Artificial Intelligence (KI'98)*, volume 1504 of *Lecture Notes in Computer Science*, pages 129–140. Springer, 1998.

[BK99] Franz Baader and Ralf Küsters. Matching in description logics with existential restrictions. In *Proc. of the 1999 Description Logic Workshop (DL'99)*. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-22/`, 1999.

[BK00] Alexander Borgida and Ralf Küsters. What's not in a name: Some properties of a purely structural approach to integrating large DL knowledge bases. In *Proc. of the 2000 Description Logic Workshop (DL 2000)*, pages 65–78. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-33/`, 2000.

[BKBM99] Franz Baader, Ralf Küsters, Alex Borgida, and Deborah L. McGuinness. Matching in description logics. *J. of Logic and Computation*, 9(3):411–447, 1999.

[BKM99] Franz Baader, Ralf Küsters, and Ralf 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.

[BKM00] Franz Baader, Ralf Küsters, and Ralf Molitor. Rewriting concepts using terminologies. In *Proc. of the 7th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2000)*, pages 297–308, 2000.

[BKT02a] Sebastian Brandt, Ralf Küsters, and Anni-Yasmin Turhan. Approximating $\mathcal{ALCN}$-concept descriptions. In *Proc. of the 2002 Description Logic Workshop (DL 2002)*, 2002.

[BKT02b] Sebastian Brandt, Ralf Küsters, and Anni-Yasmin Turhan. Approximation and difference in description logics. In D. Fensel, F. Giunchiglia, D. McGuiness, and M.-A. Williams, editors, *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR2002)*, pages 203–214, San Francisco, CA, 2002. Morgan Kaufman.

[BL84] 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.

[Bla93] Patrick Blackburn. Nominal tense logic. *Notre Dame J. of Formal Logic*, 34(1):56–83, 1993.

[BM01] Paul V. Biron and Ashok Malhotra. XML schema part 2: Datatypes. W3C Recommendation, May 2001.

[BM02] Jean-Francois Baget and Marie-Laure Mugnier. Extensions of simple conceptual graphs: the complexity of rules and constraints. *Journal of Artificial Intelligence research (JAIR)*, 16:425–465, 2002.

[BN97] Franz Baader and Paliath Narendran. Unification of concept terms in description logics. In *Proc. of the 1997 Description Logic Workshop (DL'97)*, pages 34–38, 1997.

[BN03] Franz Baader and Werner Nutt. Basic description logics. In Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 43–95. Cambridge University Press, 2003.

[Bra04] S. Brandt. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In R. López de Mantáras and L. Saitta, editors, *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI-2004)*, pages 298–302. IOS Press, 2004.

[BS85] Ronald J. Brachman and James G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[BS93] Patrick Blackburn and Edith Spaan. A modal perspective on computational complexity of attribute value grammars. *J. of Logic, Language and Information*, 2:129–169, 1993.

[BS99] Franz Baader and Ulrike Sattler. Expressive number restrictions in description logics. *J. of Logic and Computation*, 9(3):319–350, 1999.

[Bul70] R. Bull. An approach to tense logic. *Theoria*, 12:171–182, 1970.

[Cal96a] Diego Calvanese. Finite model reasoning in description logics. In *Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'96)*, pages 292–303, 1996.

[Cal96b] Diego Calvanese. Reasoning with inclusion axioms in description logics: Algorithms and complexity. In *Proc. of the 12th Eur. Conf. on Artificial Intelligence (ECAI'96)*, pages 303–307. John Wiley & Sons, 1996.

[Cal96c] Diego Calvanese. *Unrestricted and Finite Model Reasoning in Class-Based Representation Formalisms*. PhD thesis, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", 1996. Available at `http://www.dis.uniroma1.it/pub/calvanes/thesis.ps.gz`.

[CDGL95] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Structured objects: Modeling and reasoning. In *Proc. of the 4th Int. Conf. on Deductive and Object-Oriented Databases (DOOD'95)*, volume 1013 of *Lecture Notes in Computer Science*, pages 229–246. Springer, 1995.

[CDGL97] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Conjunctive query containment in Description Logics with n-ary relations. In *Proc. of the 1997 Description Logic Workshop (DL'97)*, pages 5–9, 1997.

[CDGL98a] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. On the decidability of query containment under constraints. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98)*, pages 149–158, 1998.

[CDGL98b] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. What can knowledge representation do for semi-structured data? In *Proc. of the 15th Nat. Conf. on Artificial Intelligence (AAAI'98)*, pages 205–210, 1998.

[CDGL99a] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Modeling and querying semi-structured data. *Networking and Information Systems*, 2(2), 1999.

[CDGL99b] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI'99)*, pages 84–89, 1999.

[CDGL00] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Answering queries using views over description logics knowledge bases. In *Proc. of*

the 17th Nat. Conf. on Artificial Intelligence (AAAI 2000), pages 386–391, 2000.

[CDGL02] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. 2ATAs make DLs easy. In *Proc. of the 2002 Description Logic Workshop (DL 2002)*, pages 107–118. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-53/`, 2002.

[CDGL+04] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. What to ask to a peer: Ontology-based query reformulation. In *Proc. of the 9th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2004)*, 2004.

[CDGL+05a] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Data complexity of query answering in description logics. In *Proc. of the 2005 Description Logic Workshop (DL 2005)*. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-147/`, 2005.

[CDGL+05b] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. DL-Lite: Tractable description logics for ontologies. In *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005)*, pages 602–607, 2005.

[CG05a] B. Cuenca-Grau. OWL 1.1 Web Ontology Language Model-theoretic Semantics. *Draft Document*, 2005.

[CG05b] B. Cuenca-Grau. Tractable Fragments of the OWL 1.1 Web Ontology Language. *Draft Document*, 2005.

[CHW05] CuiMing Chen, Volker Haarslev, and JiaoYue Wang. LAS: Extending Racer by a Large ABox Store. In *Proc. of the 2005 Description Logic Workshop (DL 2005)*. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-147/`, 2005.

[CKV90] S. S. Cosmadakis, P. C. Kanellakis, and M. Vardi. Polynomial-time implication problems for unary inclusion dependencies. *J. of the ACM*, 37(1):15–46, January 1990.

[CL93] Tiziana Catarci and Maurizio Lenzerini. Representing and using interschema knowledge in cooperative information systems. *J. of Intelligent and Cooperative Information Systems*, 2(4):375–398, 1993.

[CL94a] Diego Calvanese and Maurizio Lenzerini. Making object-oriented schemas more expressive. In *Proc. of the 13th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'94)*, pages 243–254, 1994.

[CL94b] Diego Calvanese and Maurizio Lenzerini. On the interaction between ISA and cardinality constraints. In *Proc. of the 10th IEEE Int. Conf. on Data Engineering (ICDE'94)*, pages 204–213, Houston (Texas, USA), 1994. IEEE Computer Society Press.

[CLN94] Diego Calvanese, Maurizio Lenzerini, and Daniele Nardi. A unified framework for class based representation formalisms. In *Proc. of the 4th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'94)*, pages 109–120, 1994.

[CLN99] Diego Calvanese, Maurizio Lenzerini, and Daniele Nardi. Unifying class-based representation formalisms. *J. of Artificial Intelligence Research*, 11:199–240, 1999.

[CR03] Diego Calvanese and Riccardo Rosati. Answering recursive queries under keys and foreign keys is undecidable. In *Proc. of the 10th Int. Workshop on Knowledge Representation meets Databases (KRDB 2003)*, pages 3–14. CEUR Electronic Workshop Proceedings, http://ceur-ws.org/Vol-79/, 2003.

[CRP+93] R. Cote, D. Rothwell, J Palotay, R. Beckett, and L. Brochu. The systematized nomenclature of human and veterinary medicine. Technical report, SNOMED International, Northfield, IL: College of American Pathologists, 1993.

[Dan84] Ryszard Danecki. Nondeterministic Propositional Dynamic Logic with intersection is decidable. In *Proc. of the 5th Symp. on Computation Theory*, volume 208 of *Lecture Notes in Computer Science*, pages 34–53. Springer, 1984.

[dB80] Jaco de Bakker. *Mathematical Theory of Program Correctness*. Prentice-Hall, 1980.

[DG95] Giuseppe De Giacomo. *Decidability of Class-Based Knowledge Representation Formalisms*. PhD thesis, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", 1995.

[DGL94a] Giuseppe De Giacomo and Maurizio 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, 1994.

[DGL94b] Giuseppe De Giacomo and Maurizio Lenzerini. Description logics with inverse roles, functional restrictions, and n-ary relations. In *Proc. of the 4th Eur. Workshop on Logics in Artificial Intelligence (JELIA'94)*, volume 838 of *Lecture Notes in Artificial Intelligence*, pages 332–346. Springer, 1994.

[DGL94c] Giuseppe De Giacomo and Maurizio Lenzerini. On the correspondence between description logics and logics of programs (position paper). In *Proc. of the Description Logics Workshop*, pages 1–4, 1994.

[DGL95] Giuseppe De Giacomo and Maurizio Lenzerini. What's in an aggregate: Foundations for description logics with tuples and sets. In *Proc. of the 14th Int. Joint Conf. on Artificial Intelligence (IJCAI'95)*, pages 801–807, 1995.

[DGL97] Giuseppe De Giacomo and Maurizio Lenzerini. A uniform framework for concept definitions in description logics. *J. of Artificial Intelligence Research*, 6:87–110, 1997.

[DLN+98] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, Werner Nutt, and Andrea Schaerf. An epistemic operator for description logics. *Artificial Intelligence*, 100(1–2):225–274, 1998.

[DLNS94] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. Deduction in concept languages: From subsumption to instance checking. *J. of Logic and Computation*, 4(4):423–452, 1994.

[DLNS98] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. $\mathcal{AL}$-log: Integrating Datalog and description logics. *J. of Intelligent Information Systems*, 10(3):227–252, 1998.

[DMO92] Robert Dionne, Eric Mays, and Frank J. Oles. A non-well-founded approach to terminological cycles. In *Proc. of the 10th Nat. Conf. on Artificial Intelligence (AAAI'92)*, pages 761–766. AAAI Press/The MIT Press, 1992.

[DNR97] Francesco M. Donini, Daniele Nardi, and Riccardo Rosati. Autoepistemic description logics. In *Proc. of the 15th Int. Joint Conf. on Artificial Intelligence (IJCAI'97)*, pages 136–141, 1997.

[DNR02] Francesco M. Donini, Daniele Nardi, and Riccardo Rosati. Description logics of minimal knowledge and negation as failure. *ACM Trans. on Computational Logic*, 3(2):177–225, 2002.

[EGM97] Thomas Eiter, Georg Gottlob, and Heikki Mannilla. Disjunctive Datalog. *ACM Trans. on Database Systems*, 22(3):364–418, 1997.

[ELST04] Thomas Eiter, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits. Combining answer set programming with description logics for the semantic web. In *Proc. of the International Conference of Knowledge Representation and Reasoning (KR'04)*, 2004.

[Eme96] E. Allen Emerson. Automated temporal reasoning about reactive systems. In Faron Moller and Graham Birtwistle, editors, *Logics for Concurrency:*

Structure versus Automata, volume 1043 of *Lecture Notes in Computer Science*, pages 41–101. Springer, 1996.

[FBDC85] M. Fattorosi-Barnaba and F. De Caro. Graded modalities I. *Studia Logica*, 44:197–221, 1985.

[Fin72] K. Fine. In so many possible worlds. *Notre Dame J. of Formal Logic*, 13(4):516–520, 1972.

[FKLS03] Enrico Franconi, Gabriel Kuper, A. Lopatenko, and L. Serafini. A robust logical and computational characterisation of peer-to-peer database systems. In *International VLDB Workshop On Databases, Information Systems and Peer-to-Peer Computing (DBISP2P'03)*, 2003.

[FL79] Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *J. of Computer and System Sciences*, 18:194–211, 1979.

[Fre95] J. W. Freeman. *Improvements to Propositional Satisfiability Search Algorithms*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, 1995.

[FvHH+01] Dieter Fensel, Frank van Harmelen, Ian Horrocks, Deborah L. McGuinness, and Peter F. Patel-Schneider. OIL: An ontology infrastructure for the Semantic Web. *IEEE Intelligent Systems*, 16(2):38–45, 2001.

[GG93] George Gargov and Valentin Goranko. Modal logic with names. *J. of Philosophical Logic*, 22:607–636, 1993.

[GH05] Birte Glimm and Ian Horrocks. Handling cyclic conjunctive queries. In *Proc. of the 2005 Description Logic Workshop (DL 2005)*, volume 147. CEUR (`http://ceur-ws.org/`), 2005.

[GHVD03] Benjamin N. Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker. Description logic programs: Combining logic programs with description logic. In *Proc. of the 12th Int. World Wide Web Conf. (WWW 2003)*, pages 48–57, 2003.

[GKWZ03] D.M. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyaschev. *Many-Dimensional Modal Logics: Theory and Applications*. Elsevier, 2003.

[GL91] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.

[GMF+03] J.H. Gennari, M.A. Musen, R.W. Fergerson, W.E. Grosse, M. Crubezy, H. Eriksson, N.F. Noy, and S.W. Tu. The evolution of protege: an environment for knowledge-based systems development. *International Journal of Human-Computer Studies*, 58(1):89–123, 2003.

[GOR97] Erich Grädel, Martin Otto, and Eric Rosen. Two-variable logic with counting is decidable. In *Proc. of the 12th IEEE Symp. on Logic in Computer Science (LICS'97)*, pages 306–317. IEEE Computer Society Press, 1997.

[GP88] George Gargov and Solomon Passy. Determinism and looping in combinatory PDL. *Theoretical Computer Science*, 61:259–277, 1988.

[Grä98] Erich 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.

[Grä99] Erich Grädel. On the restraining power of guards. *J. of Symbolic Logic*, 64:1719–1742, 1999.

[GS98] Chiara Ghidini and Luciano Serafini. Distributed first order logics. In Franz Baader and Klaus Ulrich Schulz, editors, *Frontiers of Combining Systems 2*, Berlin, 1998. Research Studies Press.

[GW04] N. Guarino and C. Welty. An overview of ontoclean. In *Handbook of Ontologies*. 2004.

[Han92] Philipp Hanschke. Specifying role interaction in concept languages. In *Proc. of the 3rd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'92)*, pages 318–329. Morgan Kaufmann, 1992.

[Har84] David Harel. Dynamic logic. In D. M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume II, pages 497–604. D. Reidel Publishing Company, 1984.

[Har85] David Harel. Recurring dominoes: Making the highly undecidable highly understandable. *Ann. of Discrete Mathematics*, 24:51–72, 1985.

[Har86] David Harel. Effective transformations of infinite trees, with applications to high undecidability, dominoes, and fairness. *J. of the ACM*, 33(1):224–248, 1986.

[Hay04] Patrick Hayes. RDF model theory. W3C Recommendation, 10 February 2004.

[HB91] Bernhard Hollunder and Franz 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.

[HH01] Jeff Heflin and James Hendler. A portrait of the Semantic Web in action. *IEEE Intelligent Systems*, 16(2):54–59, 2001.

[HKS05] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The irresistible $\mathcal{SRIQ}$. In *Proc. of the First OWL Experiences and Directions Workshop*, 2005.

[HKS06] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The even more irresistible $\mathcal{SROIQ}$. In *KR-06*, 2006. To appear.

[HKT00] David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic Logic*. The MIT Press, 2000.

[HLPT91] Berhnard Hollunder, Armin Laux, Hans-Jürgen Profitlich, and Th. Trenz. $\mathcal{KRIS}$-manual. Technical report, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), 1991.

[HLTB04] Ian Horrocks, Lei Li, Daniele Turi, and Sean Bechhofer. The Instance Store: DL reasoning with large numbers of individuals. In *Proc. of the 2004 Description Logic Workshop (DL 2004)*. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-104/`, 2004.

[HM00] Volker Haarslev and Ralf Möller. Expressive ABox reasoning with number restrictions, role hierarchies, and transitively closed roles. In *Proc. of the 7th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2000)*, pages 273–284, 2000.

[HM01a] Volker Haarslev and Ralf Möller. Combining tableaux and algebraic methods for reasoning with qualified number restrictions. In *Proc. of the 2001 Description Logic Workshop (DL 2001)*, pages 152–161. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-49/`, 2001.

[HM01b] Volker Haarslev and Ralf Möller. Optimizing reasoning in description logics with qualified number restrictions. In *Proc. of the 2001 Description Logic Workshop (DL 2001)*, pages 142–151. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-49/`, 2001.

[HM01c] Volker Haarslev and Ralf Möller. RACER system description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, volume 2083 of *Lecture Notes in Artificial Intelligence*, pages 701–705. Springer, 2001.

[HM04] Volker Haarslev and Ralf Möller. Optimization techniques for retrieving resources described in OWL/RDF documents: First results. In *Proc. of the 9th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2004)*, 2004.

[HMS04] Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Reducing $\mathcal{SHIQ}$-description logic to disjunctive datalog programs. In *Proc. of the 9th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2004)*, pages 152–162, 2004.

[HMS05] Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Data complexity of reasoning in very expressive description logics. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, pages 466–471, 2005.

[HMT01] Volker Haarslev, Ralf Möller, and Anni-Yasmin Turhan. Exploiting pseudo models for TBox and ABox reasoning in expressive description logics. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, volume 2083 of *Lecture Notes in Artificial Intelligence*. Springer, 2001.

[HMW01] Volker Haarslev, Ralf Möller, and Michael Wessel. The description logic $\mathcal{ALCNH}_{R+}$ extended with concrete domains: A practically motivated approach. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, pages 29–44, 2001.

[HN90] Bernhard Hollunder and Werner Nutt. Subsumption algorithms for concept languages. Technical Report RR-90-04, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Kaiserslautern (Germany), 1990.

[Hol94] Bernhard Hollunder. *Algorithmic Foundations of Terminological Knowledge Representation Systems*. PhD thesis, University of Saarbrücken, Department of Computer Science, 1994.

[Hor97] Ian Horrocks. *Optimising Tableaux Decision Procedures for Description Logics*. PhD thesis, University of Manchester, 1997.

[Hor98a] 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.

[Hor98b] Ian Horrocks. The FaCT system. In Harrie de Swart, editor, *Proc. of the 7th 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, 1998.

[Hor98c] Ian 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.

[HPS04a] Ian Horrocks and Peter Patel-Schneider. Reducing OWL entailment to description logic satisfiability. *J. of Web Semantics*, 1(4), 2004.

[HPS04b] Ian Horrocks and Peter F. Patel-Schneider. A proposal for an OWL rules language. In *Proc. of WWW*, 2004.

[HPSvH03] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *J. of Web Semantics*, 1(1):7–26, 2003.

[HS01] Ian Horrocks and Ulrike Sattler. Ontology reasoning in the $\mathcal{SHOQ}$(D) description logic. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 199–204, 2001.

[HS02] Ian Horrocks and Ulrike Sattler. Optimised reasoning for $\mathcal{SHIQ}$. In *Proc. of the 15th Eur. Conf. on Artificial Intelligence (ECAI 2002)*, pages 277–281, July 2002.

[HS03] Ian Horrocks and Ulrike Sattler. Decidability of $\mathcal{SHIQ}$ with complex role inclusion axioms. In *Proc. of IJCAI*, 2003.

[HS05] Ian Horrocks and Ulrike Sattler. A tableaux decision procedure for $\mathcal{SHOIQ}$. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, pages 448–453, 2005.

[HST99] Ian Horrocks, Ulrike Sattler, and Stephan 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, 1999.

[HST00a] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for very expressive description logics. *J. of the Interest Group in Pure and Applied Logic*, 8(3):239–264, 2000.

[HST00b] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Reasoning with individuals for the description logic $\mathcal{SHIQ}$. In David McAllester, editor, *Proc. of the 17th Int. Conf. on Automated Deduction (CADE 2000)*, volume 1831 of *Lecture Notes in Computer Science*, pages 482–496. Springer, 2000.

[HT00a] Ian Horrocks and Sergio Tessaris. A conjunctive query language for description logic ABoxes. In *Proc. of the 17th Nat. Conf. on Artificial Intelligence (AAAI 2000)*, pages 399–404, 2000.

[HT00b] Ian Horrocks and Stephan Tobies. Reasoning with axioms: Theory and practice. In *Proc. of the 7th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2000)*, pages 285–296, 2000.

[Hul88] Richard Hull. A survey of theoretical research on typed complex database objects. In J. Paredaens, editor, *Databases*, pages 193–256. Academic Press, 1988.

[KM01] Ralf Küsters and Ralf Molitor. Approximating most specific concepts in description logics with existential restrictions. In Franz Baader, Gerd Brewka, and Thomas Eiter, editors, *Proc. of the Joint German/Austrian Conf. on Artificial Intelligence (KI 2001)*, volume 2174 of *Lecture Notes in Artificial Intelligence*, pages 33–47. Springer, 2001.

[Koz83] Dexter Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[KPCG05] A. Kalyanpur, B. Parsia, and B. Cuenca-Grau. Beyond asserted axioms: Fine-grain justifications for owl-dl entailments. In *Proc. of the 2005 International Workshop on Description Logics*, 2005.

[KPS+05] A. Kalyanpur, Bijan Parsia, Evren Sirin, Bernardo Cuenca-Grau, and James Hendler. SWOOP: a web ontology editing browser. *J. of Web Semantics*, 4(2), 2005.

[KPSH05] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, and James Hendler. Debugging unsatisfiable classes in OWL ontologies. *J. of Web Semantics – Special Issue on the Semantic Web Track of WWW 2005*, 3(4), 2005.

[KT90] Dexter Kozen and Jerzy Tiuryn. Logics of programs. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science — Formal Models and Semantics*, pages 789–840. Elsevier Science Publishers, 1990.

[Küs98] Ralf Küsters. Characterizing the semantics of terminological cycles in $\mathcal{ALN}$ using finite automata. In *Proc. of the 6th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'98)*, pages 499–510, 1998.

[KV93] Gabriel M. Kuper and Moshe Y. Vardi. On the complexity of queries in the logical data model. *Theoretical Computer Science*, 116:33–58, 1993.

[Lam96] Patrick Lambrix. *Part-Whole Reasoning in Description Logic*. PhD thesis, Dep. of Computer and Information Science, Linköping University, Sweden, 1996.

[Len02] Maurizio Lenzerini. Data integration: A theoretical perspective. In *Proc. of the 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2002)*, pages 233–246, 2002.

[LLMW06] Hongkai Liu, Carsten Lutz, Maya Milicic, and Frank Wolter. Description logic actions with general TBoxes: a pragmatic approach. In *Proceedings of the 2006 International Workshop on Description Logics (DL2006)*, 2006.

[Llo87] J. W. Lloyd. *Foundations of logic programming; (2nd extended ed.)*. Springer-Verlag New York, Inc., New York, NY, USA, 1987.

[LM05] Carsten Lutz and Maja Milicic. A tableaux algorithm for description logics with concrete domains and gcis. In *Proc. of the 14th Int. Conf. on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX 2005)*, pages 201–216, 2005.

[LN04] T. Liebig and O. Noppens. OntoTrack: Combining browsing and editing with reasoning and explaining for OWL ontologies. In *Proc. of ISWC*, 2004.

[LR89] Christophe Lecluse and Philippe Richard. Modeling complex structures in object-oriented databases. In *Proc. of the 8th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'89)*, pages 362–369, 1989.

[LR98] Alon Y. Levy and Marie-Christine Rousset. Combining Horn rules and description logics in CARIN. *Artificial Intelligence*, 104(1–2):165–209, 1998.

[LS91] Maurizio Lenzerini and Andrea Schaerf. Concept languages as query languages. In *Proc. of the 9th Nat. Conf. on Artificial Intelligence (AAAI'91)*, pages 471–476, 1991.

[LS01] Carsten Lutz and Ulrike Sattler. The complexity of reasoning with boolean modal logics. In F. Wolter, H. Wansing, M. de Rijke, and M. Zakharyaschev, editors, *Advances in Modal Logics*, volume 3. CSLI Publications, 2001.

[LST99] Carsten Lutz, Ulrike Sattler, and Stephan Tobies. A suggestion for an n-ary description logic. In *Proc. of the 1999 Description Logic Workshop (DL'99)*, pages 81–85. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-22/`, 1999.

[LST05] C. Lutz, U. Sattler, and L. Tendera. The complexity of finite model reasoning in description logics. *Information and Computation*, 199:132–171, 2005.

[Lut99a] Carsten Lutz. Complexity of terminological reasoning revisited. In *Proc. of the 6th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR'99)*, volume 1705 of *Lecture Notes in Artificial Intelligence*, pages 181–200. Springer, 1999.

[Lut99b] Carsten Lutz. Reasoning with concrete domains. In *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI'99)*, pages 90–95, 1999.

[Lut01a] Carsten Lutz. Interval-based temporal reasoning with general TBoxes. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 89–94, 2001.

[Lut01b] Carsten Lutz. NEXPTIME-complete description logics with concrete domains. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, volume 2083 of *Lecture Notes in Artificial Intelligence*, pages 45–60. Springer, 2001.

[Lut03] Carsten Lutz. Description logics with concrete domains: A survey. In Philippe Balbiani, Nobu-Yuki Suzuki, Frank Wolter, and Michael Zakharyaschev, editors, *Advances in Modal Logics*, volume 4. King's College Publications, 2003.

[Lut04] Carsten Lutz. NExpTime-complete description logics with concrete domains. *ACM Transactions on Computational Logic*, 5(4):669–705, 2004.

[Lut05] Carsten Lutz. Pdl with intersection and converse is decidable. LTCS-Report 05-05, Technical University Dresden, 2005. Available from http://lat.inf.tu-dresden.de/research/reports.html.

[Mar99] Maarten Marx. Complexity of products of modal logics. *J. Log. Comput.*, 9(2):197–214, 1999.

[McG96a] D. McGuinness. *Explaining Reasoning in Description Logics*. PhD thesis, New Brunswick, New Jersey, 1996.

[McG96b] Deborah L. McGuinness. *Explaining Reasoning in Description Logics*. PhD thesis, Department of Computer Science, Rutgers University, October 1996. Also available as Rutgers Technical Report Number LCSR-TR-277.

[MH03] Ralf Möller and Volker Haarslev. Description logic systems. In Baader et al. [BCM+03], chapter 8, pages 282–305.

[MHW06] Ralf Möller, Volker Haarslev, and Michael Wessel. On the scalability of description logic instance retrieval. In Chr. Freksa, Kohlhase, and K. M. Schill, editors, *29. Deutsche Jahrestagung für Künstliche Intelligenz*, Lecture Notes in Artificial Intelligence. Springer Verlag, 2006. forthcoming.

[Möl00] Ralf Möller. Expressive description logics: Foundations for applications, 2000. Habilitation Thesis.

[Mot05] Boris Motik. On the properties of metamodeling in OWL. In *Proc. of ISWC*, 2005.

[Mot06] Boris Motik. *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, Univ. Karlsruhe, 2006.

[MSH04] Boris Motik, Ulrike Stattler, and Ullrich Hustadt. Reducing SHIQ description logic to disjunctive datalog programs. In *Proc. of the International Conference of Knowledge Representation and Reasoning (KR'04)*, 2004.

[MSS04] B. Motik, U. Sattler, and R. Studer. Query answering in OWL-DL with rules. In *Proc. of ISWC*, 2004.

[MSS05] Boris Motik, Ulrike Sattler, and Rudi Studer. Query answering for OWL-DL with rules. *Journal of Web Semantics*, 3:41–60, 2005.

[MT91] Wiktor Marek and Miroslaw Truszczyński. Autoepistemic logic. *J. ACM*, 38(3):587–618, 1991.

[Neb88] Bernhard Nebel. Computational complexity of terminological reasoning in BACK. *Artificial Intelligence*, 34(3):371–383, 1988.

[Neb90] Bernhard Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43:235–249, 1990.

[Neb91] Bernhard Nebel. Terminological cycles: Semantics and computational properties. In John F. Sowa, editor, *Principles of Semantic Networks*, pages 331–361. Morgan Kaufmann, 1991.

[OCE06] Maria Magdalena Ortiz, Diego Calvanese, and Thomas Eiter. Characterizing data complexity for conjunctive query answering in expressive description logics. In *Proc. of the 21st Nat. Conf. on Artificial Intelligence (AAAI 2006)*, 2006.

[OK99] Hans Jurgen Ohlbach and Jana Koehler. Modal logics, description logics and arithmetic reasoning. *Artificial Intelligence*, 109(1-2):1–31, 1999.

[Par70] David Park. Fixpoint induction and proofs of program properties. *Machine Intelligence*, 5:59–78, 1970.

[Par76] David Park. Finiteness is mu-ineffable. *Theoretical Computer Science*, 3:173–181, 1976.

[PH05a] Jeff Pan and Ian Horrocks. OWL-Eu: Adding customised datatypes into OWL. In *Proc. of ESWC*. 2005.

[PH05b] Ian Pratt-Hartmann. Complexity of the two-variable fragment with counting quantifiers. *Journal of Logic, Language, and Information*, 14(3):369–395, 2005.

[PL94] Lin Padgham and Patrick Lambrix. A framework for part-of hierarchies in terminological logics. In *Proc. of the 4th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'94)*, pages 485–496, 1994.

[Pra79] Vaugham R. Pratt. Models of program logic. In *Proc. of the 20th Annual Symp. on the Foundations of Computer Science (FOCS'79)*, pages 115–122, 1979.

[Pri67] Arthur Prior. *Past, Present, and Future*. Oxford University Press, 1967.

[PS98] P. F. Patel-Schneider. DLP system description. In *Proc. of the 1998 Description Logic Workshop (DL'98)*, pages 87–89. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-11/`, 1998.

[PS05] Peter Patel-Schneider. Building the semantic web tower from RDF straw. In *Proc. of IJCAI*, 2005.

[PS06] Eric Prud'hommeaux and Andy Seaborne. SPARQL query language for RDF. W3C Candidate Recommendation, 6 April 2006.

[PSHH04] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. OWL web ontology language semantics and abstract syntax. W3C Recommendation, 10 February 2004.

[PT85] Solomon Passy and Tinko Tinchev. PDL with data constraints. *Information Processing Lett.*, 20:35–41, 1985.

[PT91] Solomon Passy and Tinko Tinchev. An essay in combinatory dynamic logic. *Information and Computation*, 93:263–332, 1991.

[Rec02] A. Rector. Analysis of propagation along transitive roles: Formalisation of the galen experience with medical ontologies. In *Proc. of the 2002 Description Logic Workshop*, 2002.

[Rei92] Raymond Reiter. What should a database know? *Journal of Logic Programming*, 14(2,3), 1992.

[RH97] A. Rector and I. Horrocks. Experience building a large, re-usable medical ontology using a description logic with transitivity and concept inclusions. In *Proceedings of the Workshop on Ontological Engineering, AAAI Spring Symposium (AAAI'97)*, Stanford, CA, 1997. AAAI Press.

[Rob71] R. Robinson. Undecidability and nonperiodicity of tilings on the plane. *Inventiones Math.*, 12:177–209, 1971.

[Ros98] Riccardo Rosati. Autoepistemic description logics. *AI Communications—The Eur. J. on Artificial Intelligence*, 11(3–4):219–221, 1998.

[Ros05] Riccardo Rosati. On the decidability and complexity of integrating ontologies and rules. *J. of Web Semantics*, 3(1), 2005.

[Ros06] Riccardo Rosati. DL+log: Tight integration of description logics and disjunctive datalog. In *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR 2006)*, 2006. To appear.

[Sat96] Ulrike Sattler. A concept language extended with different kinds of transitive roles. In Günter Görz and Steffen Hölldobler, editors, *Proc. of the 20th German Annual Conf. on Artificial Intelligence (KI'96)*, number 1137 in Lecture Notes in Artificial Intelligence, pages 333–345. Springer, 1996.

[Sat98] Ulrike Sattler. *Terminological Knowledge Representation Systems in a Process Engineering Application*. PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany, 1998.

[SC03] S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *Proceedings of IJCAI, 2003*, 2003.

[Sch89] James G. Schmolze. Terminological knowledge representation systems supporting n-ary terms. In *Proc. of the 1st Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'89)*, pages 432–443, 1989.

[Sch91] Klaus 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, 1991.

[Sch94a] Andrea Schaerf. Reasoning with individuals in concept languages. *Data and Knowledge Engineering*, 13(2):141–176, 1994.

[Sch94b] Klaus Schild. Terminological cycles and the propositional μ-calculus. In *Proc. of the 4th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'94)*, pages 509–520, 1994.

[SE89] Robert S. Streett and E. Allen Emerson. An automata theoretic decision procedure for the propositional μ-calculus. *Information and Computation*, 81:249–264, 1989.

[SGP06] Evren Sirin, Bernardo Cuenca Grau, and Bijan Parsia. From wine to water: Optimizing description logic reasoning for nominals. In *International Conference on the Principles of Knowledge Representation and Reasoning (KR-2006)*, 2006.

[Spa00] K. Spackman. Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with SNOMED-RT. *J. of the Amer. Med. Informatics Ass.*, 2000. Fall Symposium Special Issue.

[SPC+05] E. Sirin, B. Parsia, B. Cuenca Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. To appear, 2005.

[SS89] Manfred Schmidt-Schauß. Subsumption in KL-ONE is undecidable. In Ron J. Brachman, Hector J. Levesque, and Ray Reiter, editors, *Proc. of the 1st Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'89)*, pages 421–431. Morgan Kaufmann, 1989.

[SSS91] Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[Sti96] Colin Stirling. Modal and temporal logics for processes. In Faron Moller and Graham Birtwistle, editors, *Logics for Concurrency: Structure versus Automata*, volume 1043 of *Lecture Notes in Computer Science*, pages 149–237. Springer, 1996.

[Sun05] B. Suntisrivaraporn. Optimization and implementation of subsumption algorithms for the description logic $\mathcal{EL}$ with cyclic TBoxes and general concept inclusion axioms. Master thesis, TU Dresden, Germany, 2005.

[SV01] Ulrike Sattler and Moshe Y. Vardi. The hybrid μ-calculus. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, pages 76–91, 2001.

[SvRvdVM95] P.-H. Speel, F. van Raalte, P. E. van der Vet, and N. J. I. Mars. Runtime and memory usage performance of description logics. In G. Ellis, R. A. Levinson, A. Fall, and V. Dahl, editors, *Knowledge Retrieval, Use and Storage for Efficiency: Proc. of the 1st Int. KRUSE Symposium*, pages 13–27, 1995.

[Swi04] Terrance Swift. Deduction in ontologies via ASP. In *Proc. of LPNMR 2004*, pages 275–288, 2004.

[TH04] Dmitry Tsarkov and Ian Horrocks. Efficient reasoning with range and domain constraints. In *Proc. of the 2004 Description Logic Workshop*, 2004.

[TH05a] Dmitry Tsarkov and Ian Horrocks. Optimised classification for taxonomic knowledge bases. In *Proc. of the 2005 Description Logic Workshop (DL 2005)*, volume 147. CEUR (`http://ceur-ws.org/`), 2005.

[TH05b] Dmitry Tsarkov and Ian Horrocks. Ordering heuristics for description logic reasoning. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, pages 609–614, 2005.

[TH06] Dmitry Tsarkov and Ian Horrocks. FaCT++ Description Logic Reasoner: System Description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, 2006. To appear.

[The00] The Gene Ontology Consortium. Gene Ontology: Tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.

[Tob99a] Stephan Tobies. A NEXPTIME-complete description logic strictly contained in C^2. In J. Flum and M. Rodríguez-Artalejo, editors, *Proc. of the Annual Conf. of the Eur. Assoc. for Computer Science Logic (CSL'99)*, volume 1683 of *Lecture Notes in Computer Science*, pages 292–306. Springer, 1999.

[Tob99b] Stephan Tobies. On the complexity of counting in description logics. In *Proc. of the 1999 Description Logic Workshop (DL'99)*, pages 105–109. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-22/`, 1999.

[Tob99c] Stephan Tobies. A PSPACE algorithm for graded modal logic. In H. Ganzinger, editor, *Proc. of the 16th Int. Conf. on Automated Deduction (CADE'99)*, volume 1632 of *Lecture Notes in Artificial Intelligence*, pages 52–66. Springer, 1999.

[Tob01] Stephan Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany, 2001.

[Var82] Moshe Y. Vardi. The complexity of relational query languages. In *Proc. of the 14th ACM SIGACT Symp. on Theory of Computing (STOC'82)*, pages 137–146, 1982.

[Var97] Moshe Y. Vardi. Why is modal logic so robustly decidable. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, volume 31, pages 149–184. American Mathematical Society, 1997.

[Var98] Moshe Y. Vardi. Reasoning about the past with two-way automata. In *Proc. of the 25th Int. Coll. on Automata, Languages and Programming (ICALP'98)*, volume 1443 of *Lecture Notes in Computer Science*, pages 628–641. Springer, 1998.

[VdH92] Wiebe Van der Hoek. On the semantics of graded modalities. *J. of Applied Non-Classical Logics*, 2(1):81–123, 1992.

[VdHdR95] Wiebe Van der Hoek and Maarten de Rijke. Counting objects. *J. of Logic and Computation*, 5(3):325–345, 1995.

[VW86] Moshe Y. Vardi and Pierre Wolper. Automata-theoretic techniques for modal logics of programs. *J. of Computer and System Sciences*, 32:183–221, 1986.

[Wan06] T.D. Wang. Gauging ontologies and schemas by numbers. In *WWW06 Workshop on Evaluation of Ontologies for the Web*, 2006. To Appear.

[WBH+05] K. Wolstencroft, A. Brass, I. Horrocks, P. Lord, U. Sattler, D. Turi, and R. Stevens. A little semantic web goes a long way in biology. In *Proc. of ISWC*, 2005.