Description Logics: Foundations for Class-based Knowledge Representation

Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, I-00198 Roma, Italy
{calvanese, degiacomo, lenzerini}@dis.uniroma1.it

Abstract

Class-based languages express knowledge in terms of objects and classes, and have inspired a huge number of formalisms in computer science. Description logics form a family of both class-based and logic-based knowledge representation languages which allow for modeling an application domain in terms of objects, classes and relationships between classes, and for reasoning about them. This paper presents an overview of the research carried out in the last years in description logics, with the main goal of illustrating how these logics provide the foundations for class-based knowledge representation formalisms.

1 Introduction

There are several families of knowledge representation languages, including logic-based, rule-based, and class-based languages. Class-based languages express knowledge in terms of objects and classes, and have inspired a huge number of formalisms in several areas of computer science, including programming languages, database models, and software specification languages. Description logics (DLs) [24, 64, 44, 5][1] form a family of languages for modeling an application domain in terms of objects, classes and relationships between classes, and for reasoning about them. Differently from object-oriented languages used in databases and programming languages, DLs permit the specification of a domain by providing the definition of classes, and by describing classes using a rich set of logical operators. They are therefore both class-based and logic-based knowledge representation languages. Notably, when using DLs, one can specify not only the necessary conditions that objects of a given class must obey, but also the sufficient conditions for an object to belong to a certain class. This feature introduces the possibility of automatically classifying objects and class descriptions. Indeed, the definition of a class may imply that it is subsumed by, or is even equivalent to, another class.

One of the main reasoning services of a DL system is to automatically build the so-called subsumption hierarchy of classes, i.e., a graph showing all the subsumption relations between the classes of an application. If one considers that all class-based languages exploit the notion of inheritance, and this notion is based on the subsumption relation, one realizes how important this kind of relation between classes is in fields such as knowledge representation, databases, and software engineering.

This paper provides an overview of the salient characteristics of DLs. Special emphasis is given to illustrating how such logics provide a foundation for the whole family of class-based knowledge representation. Also, particular attention is given to the large body of research devoted to the design of reasoning techniques for DLs, and their computational characterization. This research work has resulted in effective automated reasoning systems based on DLs. The availability of reasoning systems, and the vast number of results on algorithms and complexity for reasoning in DLs has stimulated their usage in various domains, including planning, action representation, software engineering, information systems, databases, information integration, and the semantic web.

The paper is organized as follows. In Section 2, we formally introduce DLs, in terms of both syntax and semantics, and associated reasoning tasks. Section 3 provides a short overview of the development of research in DLs, pointing out the aspects that make these logics applicable in several interesting contexts. Section 4 emphasizes the relationship with other logics, in particular propositional dynamic logics and modal logics. Section 5 illustrates the main techniques for reasoning in DLs, delving into some details about automata-based techniques. Finally, Section 6 concludes the paper, by mentioning several aspects of the research on DLs that have not been discussed in the previous sections.

[1] http://dl.kr.org/

2 What is a Description Logic?

The basic elements of DLs are *classes* (also called *concepts*) and *roles*, which denote sets of objects and binary relations, respectively. Concept expressions and role expressions (in the following simply called *concepts* and *roles*) are formed by starting from a set of *atomic concepts* and *atomic roles*, i.e., concepts and roles denoted simply by a name, and applying concept and role *constructs*. For a comprehensive discussion on the constructs used in DLs, see [124, 50, 35, 64].

The most expressive DL that we refer to in this paper is called $\mathcal{ALCQI}_{reg}$. In such logic, concepts and roles are formed according to the following syntax:

$$\begin{array}{rcl} C, C' & \longrightarrow & A \mid \neg C \mid C \sqcap C' \mid \forall R.C \mid (\geqslant n\, Q.C) \\ Q & \longrightarrow & P \mid P^- \\ R, R' & \longrightarrow & Q \mid R \cup R' \mid R \circ R' \mid R^* \mid id(C) \end{array}$$

where A and P denote respectively atomic concepts and atomic roles, and C and R denote respectively arbitrary concepts and roles, and n denotes a positive integer. We use Q to denote *basic roles*, which for $\mathcal{ALCQI}_{reg}$ are either atomic or inverses of atomic roles.

We also use the following abbreviations to increase readability:

- $\bot$ for $A \sqcap \neg A$ (where A is any atomic concept),
- $\top$ for $A \sqcup \neg A$,
- $(\leqslant n\, R.C)$ for $\neg(\geqslant n+1\, R.C)$,
- $(= n\, R.C)$ for $(\geqslant n\, R.C) \sqcap (\leqslant n\, R.C)$,
- $C \sqcup D$ for $\neg(\neg C \sqcap \neg D)$,
- $\exists R.C$ for $\neg\forall R.\neg C$.

Let us comment on the constructs of $\mathcal{ALCQI}_{reg}$. Among the constructs used in forming concept expressions we find the basic set operators, namely set complement, intersection, and union. DLs admit a restricted form of quantification which is realized through so-called *quantified role restrictions*. A quantified role restriction is composed by a quantifier (existential or universal), a role, and a concept expression. Quantified role restrictions allow one to represent the relationships existing between the objects in two concepts. For example, one can characterize the set of objects all of whose children are male as $\forall\mathsf{child}.\mathsf{Male}$, as well as the set of objects that have at least one male child as $\exists\mathsf{child}.\mathsf{Male}$. *Number restrictions* are used to constrain the number of *fillers*, i.e., the objects that are in a certain relationship with a given object. For example, $((= 2\,\mathsf{child}.\mathsf{Male}))$ characterizes the set of parents with exactly two male children. The form used in $\mathcal{ALCQI}_{reg}$, called *qualified number restriction* [79], is a very general one. It allows one to pose restrictions on the number of objects connected through a certain role, counting only those objects that satisfy a certain condition.

In addition to concept forming constructs, $\mathcal{ALCQI}_{reg}$ provides the *inverse role* construct, which allows us to denote the inverse of a given relation, and the constructs for denoting regular expressions on basic roles.

From the semantic point of view, concepts are interpreted as subsets of a domain, and roles as binary relations over that domain. An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ over a set $\mathcal{A}$ of atomic concepts and a set $\mathcal{P}$ of atomic roles consists of a nonempty set $\Delta^{\mathcal{I}}$ (the *domain* of $\mathcal{I}$) and a function $\cdot^{\mathcal{I}}$ (the *interpretation function* of $\mathcal{I}$) that maps every atomic concept $A \in \mathcal{A}$ to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ (the set of *instances* of A) and every atomic role $P \in \mathcal{P}$ to a subset $P^{\mathcal{I}}$ of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ (the set of *instances* of P). The interpretation function can then be extended to arbitrary concepts and roles as follows[2]:

$$\begin{array}{rcl} (\neg C)^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\ (C \sqcap C')^{\mathcal{I}} & = & C^{\mathcal{I}} \cap C'^{\mathcal{I}} \\ (\exists R.C)^{\mathcal{I}} & = & \{o \in \Delta^{\mathcal{I}} \mid \exists o'.\, (o,o') \in R^{\mathcal{I}} \wedge o' \in C^{\mathcal{I}}\} \\ (\geqslant n\, R.C)^{\mathcal{I}} & = & \{o \in \Delta^{\mathcal{I}} \mid \sharp\{o' \mid (o,o') \in R^{\mathcal{I}} \\ & & \qquad \wedge\, o' \in C^{\mathcal{I}}\} \geq n \\ (P^-)^{\mathcal{I}} & = & \{(o,o') \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (o',o) \in P^{\mathcal{I}}\} \\ (R \cup R')^{\mathcal{I}} & = & (R)^{\mathcal{I}} \cup (R')^{\mathcal{I}} \\ (R \circ R')^{\mathcal{I}} & = & (R)^{\mathcal{I}} \circ (R')^{\mathcal{I}} \\ (R^*)^{\mathcal{I}} & = & \text{reflexive transitive closure of } (R)^{\mathcal{I}} \\ (id(C))^{\mathcal{I}} & = & \{(o,o) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid o \in C^{\mathcal{I}}\} \end{array}$$

Usually, in DLs, a *knowledge base* is formed by two components, traditionally called TBox and ABox. A *TBox*, expresses intensional knowledge about classes and relations, and an *ABox* expresses extensional knowledge about individual objects. Here we concentrate on intensional knowledge only, and therefore we identify a knowledge base with a TBox.

Formally, an $\mathcal{ALCQI}_{reg}$ knowledge base is constituted by a finite set of *inclusion assertions* of the form

$$C_1 \sqsubseteq C_2$$

with C_1 and C_2 arbitrary concept expressions.

The semantics of a knowledge base is specified through the notion of satisfaction of assertions. An interpretation $\mathcal{I}$ *satisfies* the assertion $C_1 \sqsubseteq C_2$ if $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$. An interpretation is a *model* of a knowledge base if it satisfies all assertions in it. A knowledge base is *satisfiable* if it admits a model.

Assertions of the form above are usually called arbitrary assertions. Special cases of assertions are also of interest. A primitive inclusion assertion is an inclusion assertion of the form $A \sqsubseteq C$, which specifies (by means of C) only *necessary* conditions for an object to be an instance of the atomic concept A. Symmetrically, an assertion $C \sqsubseteq A$ specifies

[2]We use $\sharp S$ to denote the cardinality of a set S.

a *sufficient* condition for an object to be an instance of A. An equality assertion $A \equiv C$, which corresponds to the pair of assertions $A \sqsubseteq C$ and $C \sqsubseteq A$, specifies both *necessary and sufficient* conditions for the instances of A. Equality assertions are typical of the frame systems from which DLs originate, where assertions of this kind (without cycles, see later) are used to define a taxonomy of concepts.

The basic reasoning tasks with respect to a given knowledge base that we consider, are the following:

- *Knowledge base satisfiability* is the problem of deciding whether a knowledge base $\mathcal{K}$ is *satisfiable*, i.e., whether $\mathcal{K}$ admits at least one model.
- *Concept consistency* is the problem of deciding whether a concept C is *consistent* in a KB $\mathcal{K}$, i.e., whether $\mathcal{K}$ admits a model $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \emptyset$.
- *Concept subsumption* is the problem of deciding whether a KB $\mathcal{K}$ *implies* an inclusion assertion $C_1 \sqsubseteq C_2$ (written as $\mathcal{K} \models C_1 \sqsubseteq C_2$), i.e., whether $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$ for each model $\mathcal{I}$ of $\mathcal{K}$.

The basic reasoning tasks above can be reduced to each other (provided the language over which the knowledge base is built is sufficiently expressive). We have that $\mathcal{K} \models C_1 \sqsubseteq C_2$ can be reformulated as inconsistency of $C_1 \sqcap \neg C_2$ in $\mathcal{K}$, while consistency of C in $\mathcal{K}$ can be reformulated as $\mathcal{K} \not\models C \sqsubseteq \bot$. In addition, consistency of C in $\mathcal{K}$ can be reformulated as satisfiability of the knowledge base $\mathcal{K} \cup \{\top \sqsubseteq \exists P_{new}.C\}$, where P_{new} is a newly introduced atomic role. Finally, satisfiability of a knowledge base K can be reformulated as consistency of $\top$ in $\mathcal{K}$.

3 Description Logics as a Basis for Class-based Formalisms

As we said in the introduction, one of the main research lines of the field of knowledge representation has been concerned with the idea that the knowledge structure should be expressed in terms of the classes of objects that are of interest in the domain, as well as the relevant relationships holding among such classes. Consequently, investigation of suitable formalisms for representing classes and their relationships, has been carried out since the beginning of the research on knowledge representation.

The above principle formed the basis for the development of the first frame systems and semantic networks. However, such systems were in general defined informally, and the associated reasoning tools were strongly dependent on the implementation strategies. A fundamental step towards a logic-based characterization of such systems has been accomplished through the work on the KL-ONE system [28], which collected many of the ideas stemming from earlier semantic networks and frame-based systems, and provided a logical basis for interpreting objects, classes (or concepts), and relationships (or roles) between them (see for example, [124]). A basic goal of such a logical reconstruction was the precise characterization of the set of constructs used to build class and role expressions. Providing a formal meaning to the constructs of the representation language has been fundamental, but knowledge representation systems should also come with reasoning procedures that are sound and complete with respect to a specified formal semantics, and that are precisely characterized in terms of computational complexity. With the article "The tractability of subsumption in Frame-Based Description Languages" by [26], a research line addressing the tradeoff between the expressiveness of KL-ONE like languages and the computational complexity of reasoning was originated. In fact, it was shown that an apparently minor extension of the language could make the basic deduction problem in the language computationally hard (even undecidable). [26] is now generally considered as the first paper on DLs. Many subsequent papers addressed the question of the computational complexity of reasoning in DLs in a simplified context where both the TBox and the ABox are empty [96, 112, 60, 64, 61]. This is not surprising, since these works aimed at studying the language constructs in isolation, with the goal of singling out their impact on the complexity of subsumption between concept expressions. After more than a decade, this investigation is basically completed, and we have now a global picture of the expressiveness/complexity trade-off. It is worth mentioning that these results have led to the development of successful DL systems, such as CLASSIC [27], that have been extremely important in showing the effectiveness of applying DLs in real world applications.

In the last years, the investigation on DLs has been driven by the goal of applying class-based representation formalisms in several areas, such as planning [123], action representation [2], software engineering [58], information systems [49], databases [24, 19, 113], information integration [45], intelligent access to the web [91, 23], semantic web [69].

The modeling requirements arising in the above areas have stimulated the need of incorporating increasingly expressive representation mechanisms, and to adapt/extend the corresponding reasoning techniques:

- The goal of capturing the semantics of database models and reasoning about data schemas has stressed the importance of number restrictions, and cyclic assertions in the knowledge base [47, 48]. Similar requirements are emerged with the need of representing ontologies in the context of the semantic web [82].
- Information integration systems require inclusion assertions not only on concepts, but also on relations [118, 45, 75].

- Semistructured data, used in applications such as digital libraries, Internet information systems, etc., require the ability to represent data whose structure is not rigid and strictly typed as in conventional database systems. Models for semi-structured data represent data as graphs with labeled edges, and adopt flexible typing schemes in order to classify data [33]. A special case of such models is XML [29], which is becoming the standard for exchanging data on the web. In general, correctly modeling such typing schemes calls for the use of fixpoints in the representation formalism [38, 40].

- UML [106] is nowadays the standard language for the analysis phase of software and information system development. CASE tools that perform automated reasoning on UML schemas (for example, to test consistency or redundancy) would be of great interest. Fully capturing UML schemas in DLs requires inverse roles, number restrictions, and general fixpoints on concepts for modeling recursive structures [39].

Virtually all the above characteristics point out the need for dealing with cyclic assertions in the knowledge base. The presence of a cycles in the assertions of a knowledge base does have a strong impact on the DL. Different types of semantics can be adopted, which differ in the interpretation of cycles (but coincide for acyclic knowledge bases). The semantics specified in the previous section is called *descriptive semantics*. Alternatively, *fixpoint semantics* have been considered, in which the assertions are viewed as equations and only those interpretations that are (least or greatest) fixpoints of the equations are accepted as models. For a detailed discussion on the different semantics, see [98, 31, 30, 111, 56].

The importance of dealing with arbitrary assertions motivated the strong interest in the problem of reasoning with TBox assertions without the acyclicity assumption [98, 3, 111, 51, 36, 81, 84]. One important outcome of this line of research is that, with this new feature, DLs have been shown to provide a unifying formalism for class-based knowledge representation [48]. Note that, when cycles are permitted, limiting the expressive power of the language with the goal of gaining tractability is useless, because the power of TBox assertions alone generally leads to high complexity in the inference mechanisms even in simple languages. For this reason, the investigation on DLs has turned towards very powerful languages for expressing concepts and roles, where the property of interest is no longer tractability of reasoning, but rather decidability [31, 35, 50].

4 Relationship with other Logics

In 1991, a seminal paper [110] pointed out a tight correspondence between DLs and propositional dynamic logics (PDLs)[3], which are modal logics specifically designed for reasoning about program schemes [86]. In particular, [110] showed 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 $\mathcal{ALCI}_{reg}$.[4]

The correspondence is based on the similarity between the interpretation structures of the two kinds of logics: at the extensional level, objects in DLs correspond to states in PDLs, whereas links between two objects 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 DLs concepts and roles to PDLs formulae and programs, respectively. In particular the mapping τ from $\mathcal{ALCI}_{reg}$ to *converse* PDL is defined inductively as follows:

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

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

One of the most important contributions of the correspondence is the so-called internalization theorem, which 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 [110]. 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 DLs tells us that for such DLs reasoning with TBoxes, i.e., logical implication, is no harder that reasoning with a single concept.

As a consequence of the correspondence, the large body of research on PDLs, and more generally modal logics, has been exploited in the context of DLs. E.g., the correspondence with modal mu-calculus [111, 52, 56] has settled the issue of which semantics to adopt for cyclic knowledge bases. Conversely, the work on DLs has lead to a number of interesting extensions of PDLs in terms of those constructs that are typical of DLs and but not well-studied in PDLs.

[3]We use the term "propositional dynamic logics" in a slightly more general sense than usual, so as to include the basic multi-modal logic K_i, and modal mu-calculus.

[4]In fact, the decidability of $\mathcal{ALC}_{reg}$ without the $id(C)$ construct was independently established in [3].

Several constructs typical of DLs have a natural counterpart in PDLs. Functional restrictions on roles [51] used in DLs are closely related to deterministic programs in PDLs [16]. Observe however that a functional restriction would impose that a program is deterministic (has a single execution) locally, i.e., the program is deterministic if it starts from a state where the functional restriction holds.

Qualified number restrictions on roles [80, 54, 55] correspond directly to *graded modalities* studied in modal logics [119, 120, 68, 70]. Note that first results on graded modalities in full-fledged PDLs were obtained from the correspondence with DLs. More recently, in the context of DLs, mu-calculus extended with inverses and graded modalities has also been studied [39].

In DLs ways to denote single objects are often considered. *ABoxes*, i.e., collections of membership assertions (see, e.g., [108, 55]), and constructs involving single individuals, such as ONE-OF or FILLS [109], are the most common ones. Recently, the notion of *object name*, i.e., a concept that has a single instance, has attracted interest in DLs [50, 116]. These names corresponds to *nominals* in modal logics [22, 73, 21, 105, 32]. PDLs with nominals are sometimes called *combinatory PDLs* [99, 74, 100]. The results on names obtained in the context of DLs [50] closed some open problems related to combinatory PDLs, by characterizing the computational complexity of *deterministic combinatory* PDL , and establishing the decidability and characterizing the computational complexity of *converse combinatory* PDL . Modal mu-calculus extended with nominals has been investigated in the context of DLs [107]. Finally the DLs that include ways to denote both a universal modality and nominals, such as those in [50, 107], are tightly related to so-called *hybrid logics*, a family of logics that has attracted the interest of modal logicians recently [1].

It is worth mentioning that, besides the above results, the correspondence between DLs and modal logics is generating quite interesting works, such as [88] on expressiveness characterization of DLs that are not propositionally closed, or [76] that looks at the guarded fragments as forms of DLs with n-ary relations, or [12] on very general conditions for combining DLs constructs without losing decidability, just to mention a few.

5 Reasoning Techniques

The study of suitable techniques for solving the reasoning problems in DLs has been developed starting with severe restrictions on the expressiveness of the language and on the form of the knowledge base. The reasoning techniques have then evolved over time, from specialized, adhoc methods to fully general ones. We discuss the various approaches that have been proposed.

5.1 Structural Techniques

The first approaches were developed under the assumption that one can embody the knowledge represented in the terminology directly into concept expressions, rather than assertions. Therefore, subsumption on concept expressions was regarded as the basic reasoning task. The proposed techniques were based on *structural comparisons* concept expressions in simple languages [26, 102, 96, 97]. At the heart of structural comparison is the idea that, if the two concept expressions to be compared are made of subexpressions, one can compare separately one subexpression of a concept with all those of the others. For example, the structural algorithm for subsumption checking in [26, 90], which applies to the subset of $\mathcal{ALCQI}_{reg}$ comprising only the constructs $C \sqcap C'$, $\forall P.C$, and $\exists P.\top$, works in two phases: First, concepts are rewritten in a normal form consisting of a conjunction of atomic concepts, concepts of the form $\exists P.\top$ and, for each role P, at most one concept of the form $\forall P.C'$, where C' is again normalized. Then, the structures of the normalized concepts are compared, which can be done by considering one conjunct from each concept at a time. The algorithm works in time linear in the size of the two concepts (provided subexpressions are ordered). The algorithm can be extended straightforwardly with the same complexity also to a few constructs, e.g., atomic negation. For more expressive languages, however, structural techniques fail since they do not capture the complex interaction that may occur between different constructs. Exceptions, obtained by considering particular combinations of constructs are reported in [101, 97].

5.2 Tableaux-based Techniques

Structural subsumption techniques are applicable only to relatively weak DLs, and only for reasoning on concept expressions, rather than assertions in a knowledge base. In [112] the notion of *constraint system* was proposed as a general technique to reason in more expressive DLs. Subsequent investigations showed that constraint systems can be seen as specialized forms of *tableaux*. Many results on algorithms for reasoning on concept expressions, and their complexity were then derived using tableau-based techniques [60, 80, 59, 63, 61]. Such techniques, besides being intuitively appealing, provided a useful framework for modularizing the problem of designing reasoning algorithms for languages formed by different sets on constructs. In fact, a tableau-based algorithm essentially amounts to providing an expansion rule for each of the constructs in the language, and then showing the correctness of each rule and the termination of the expansion process. The algorithms for concept satisfiability and subsumption obtained in this way have also lead to actual implementations, by application of clever control strategies and optimization techniques [96, 103, 7].

In [4, 7, 31], tableau-based techniques for reasoning on arbitrary inclusion assertions have been proposed. Such techniques [13] have later been refined and optimized, and have led to the development of various systems for reasoning in expressive DLs in the presence of assertions [81, 84, 83, 78, 77].

Tableaux techniques can also be used in the presence of regular expressions over roles [3] (see also [104, 57] in the context of PDLs). However, current DL reasoning systems still do not support regular expressions over roles. Moreover, to obtain tableaux algorithms that are computationally tight (i.e., EXPTIME-complete) in the presence of arbitrary inclusion assertions, requires quite sophisticated techniques [66], which are not implemented in actual systems.

5.3 Automata-based Techniques

The most successful techniques for reasoning in expressive DLs that include forms of fixpoints (regular expressions, well-foundedness constraints, fixpoint constructs) are automata-theoretic. In particular, the correspondence between PDLs and DLs described in Section 4, has led to the application, also in the context of DLs, of techniques based on automata on infinite trees. Such techniques, which rely on the *tree-model property* shared by most PDLs (and DLs), have been used to devise computationally optimal reasoning procedures for virtually all variants of expressive PDLs [114, 115, 122, 121]. Recently, they have been extended and adapted to take into account typical constructs of DLs, such as qualified number restrictions and object names [39, 107, 43].

Here we give some detail on techniques based on *two-way alternating automata on infinite trees* (2ATAs), introduced originally in [121] for reasoning in the modal mu-calculus with converse. 2ATAs provide a very elegant and effective formal tool for addressing reasoning in expressive DLs. In particular, differently from usual (one-way nondeterministic) tree automata [115], they provide an high level description of the automaton computation that abstracts from the combinatorics and allows one to concentrate on the logical aspects. As a result, the encoding of a DL concept (to be checked for satisfiability) into an automaton (to be checked for non-emptiness) is intuitive (indeed, comparable to tableaux rules), modular (since each construct is dealt with separately), short (since the encoding is polynomial), and computationally adequate (i.e., optimal w.r.t. the complexity class of reasoning).

Most of the recent results on reasoning in expressive DLs can be easily reconstructed using techniques based on 2ATAs. Here we illustrate how to reduce $\mathcal{ALCFI}_{reg}$ concept satisfiability (and hence, in virtue of internalization, all other reasoning services) to non-emptiness of 2ATAs. $\mathcal{ALCFI}_{reg}$ is a restrictions of $\mathcal{ALCQI}_{reg}$ in which the only allowed number restrictions are of the form $(\leqslant\ 1\,Q.\top)$, which we abbreviate with $(\leqslant\ 1\,Q)$. The construction for $\mathcal{ALCFI}_{reg}$ can be easily extended to deal with other constructs and illustrates the main ideas of such a reduction.

Infinite trees are represented as prefix closed (infinite) sets of words over $\mathbb{N}$ (the set of positive natural numbers). Formally, an *infinite tree* is a set of words $T \subseteq \mathbb{N}^*$, such that if $x{\cdot}c \in T$, where $x \in \mathbb{N}^*$ and $c \in \mathbb{N}$, then also $x \in T$. The elements of T are called *nodes*, the empty word ε is the *root* of T, and for every $x \in T$, the nodes $x{\cdot}c$, with $c \in \mathbb{N}$, are the *successors* of x. By convention we take $x{\cdot}0 = x$, and $x{\cdot}i{\cdot}{-1} = x$. The *branching degree* $d(x)$ of a node x denotes the number of successors of x. If the branching degree of all nodes of a tree is bounded by k, we say that the tree has branching degree k. An *infinite path* P of T is a prefix-closed set $P \subseteq T$ such that for every $i \geq 0$ there exists a unique node $x \in P$ with $|x| = i$. A *labeled tree* over an alphabet Σ is a pair (T, V), where T is a tree and $V : T \to \Sigma$ maps each node of T to an element of Σ.

Alternating automata on infinite trees are a generalization of nondeterministic automata on infinite trees, introduced in [95]. They allow for an elegant reduction of decision problems for temporal and program logics [67, 20]. Let $\mathcal{B}(I)$ be the set of positive boolean formulae over I, built inductively by applying $\wedge$ and $\vee$ starting from **true**, **false**, and elements of I. For a set $J \subseteq I$ and a formula $\varphi \in \mathcal{B}(I)$, we say that J *satisfies* φ if and only if, assigning **true** to the elements in J and **false** to those in $I \setminus J$, makes φ true. For a positive integer k, let $[k] = \{-1, 0, 1, \ldots, k\}$. A *two-way alternating tree automaton* (2ATA) running over infinite trees with branching degree k, is a tuple $\mathbf{A} = \langle \Sigma, Q, \delta, q_0, F \rangle$, where Σ is the input alphabet, Q is a finite set of states, $\delta : Q \times \Sigma \to \mathcal{B}([k] \times Q)$ is the transition function, $q_0 \in Q$ is the initial state, and F specifies the acceptance condition.

The transition function maps a state $q \in Q$ and an input letter $\sigma \in \Sigma$ to a positive boolean formula over $[k] \times Q$. Intuitively, if $\delta(q, \sigma) = \varphi$, then each pair (c, q') appearing in φ corresponds to a new copy of the automaton going to the direction suggested by c and starting in state q'. For example, if $k = 2$ and $\delta(q_1, \sigma) = (1, q_2) \wedge (1, q_3) \vee (-1, q_1) \wedge (0, q_3)$, when the automaton is in the state q_1 and is reading the node x labeled by the letter σ, it proceeds either by sending off two copies, in the states q_2 and q_3 respectively, to the first successor of x (i.e., $x{\cdot}1$), or by sending off one copy in the state q_1 to the predecessor of x (i.e., $x{\cdot}{-1}$) and one copy in the state q_3 to x itself (i.e., $x{\cdot}0$).

A run of a 2ATA $\mathbf{A}$ over a labeled tree (T, V) is a labeled tree (T_r, r) in which every node is labeled by an element of $T \times Q$. A node in T_r labeled by (x, q) describes a copy of $\mathbf{A}$ that is in the state q and reads the node x of T. The labels of adjacent nodes have to satisfy the transition function of $\mathbf{A}$. Formally, a run (T_r, r) is a $T \times Q$-labeled tree satisfying:

1. $\varepsilon \in T_r$ and $r(\varepsilon) = (\varepsilon, q_0)$.

2. Let $y \in T_r$, with $r(y) = (x, q)$ and $\delta(q, V(x)) = \varphi$. Then there is a (possibly empty) set $S = \{(c_1, q_1), \ldots, (c_n, q_n)\} \subseteq [k] \times Q$ such that:

   - S satisfies φ and
   - for all $1 \leq i \leq n$, we have that $y{\cdot}i \in T_r$, $x{\cdot}c_i$ is defined, and $r(y{\cdot}i) = (x{\cdot}c_i, q_i)$.

A run (T_r, r) is *accepting* if all its infinite paths satisfy the acceptance condition[5]. Given an infinite path $P \subseteq T_r$, let $\mathit{inf}(P) \subseteq Q$ be the set of states that appear infinitely often in P (as second components of node labels). We consider here *Büchi* acceptance conditions. A Büchi condition over a state set Q is a subset F of Q, and an infinite path P satisfies F if $\mathit{inf}(P) \cap F \neq \emptyset$.

The non-emptiness problem for 2ATAs consists in determining, for a given 2ATA, whether the set of trees it accepts is nonempty. In [121] it is shown that deciding nonemptiness of a 2ATA with n states and an input alphabet with m elements can be done in time exponential in n and polynomial in m.

We show how to decide satisfiability of $\mathcal{ALCFI}_{reg}$ concepts by reducing it to nonemptiness of 2ATAs. To this end we first define the *(syntactic) closure* for $\mathcal{ALCFI}_{reg}$, which extends the standard Fischer-Ladner for *converse* PDL [71], by treating functional restrictions as atomic concepts. For technical reasons we include in the closure also additional elements representing basic roles and their negations. In particular, the closure $CL_{\mathcal{F}}(C_0)$ of an $\mathcal{ALCFI}_{reg}$ concept C_0 is defined as the smallest set of concepts such that $C_0 \in CL_{\mathcal{F}}(C_0)$ and such that (assuming $\sqcup$ and $\forall$ to be expressed by means of $\sqcap$ and $\exists$, and using Q^- to denote P when $Q = P^-$)[6]:

$$\begin{array}{lll}
C \in CL_{\mathcal{F}}(C_0) & \Rightarrow & \neg C \in CL_{\mathcal{F}}(C_0) \\
 & & \text{(if } C \text{ is not of the form } \neg C'\text{)} \\
\neg C \in CL_{\mathcal{F}}(C_0) & \Rightarrow & C \in CL_{\mathcal{F}}(C_0) \\
C \sqcap C' \in CL_{\mathcal{F}}(C_0) & \Rightarrow & C,\, C' \in CL_{\mathcal{F}}(C_0) \\
\exists R.C \in CL_{\mathcal{F}}(C_0) & \Rightarrow & C \in CL_{\mathcal{F}}(C_0) \\
\exists (R \cup R').C \in CL_{\mathcal{F}}(C_0) & \Rightarrow & \exists R.C,\, \exists R'.C \in CL_{\mathcal{F}}(C_0) \\
\exists (R \circ R').C \in CL_{\mathcal{F}}(C_0) & \Rightarrow & \exists R.\exists R'.C \in CL_{\mathcal{F}}(C_0) \\
\exists R^*.C \in CL_{\mathcal{F}}(C_0) & \Rightarrow & \exists R.\exists R^*.C \in CL_{\mathcal{F}}(C_0) \\
\exists id(C).C' \in CL_{\mathcal{F}}(C_0) & \Rightarrow & C \in CL_{\mathcal{F}}(C_0)
\end{array}$$

$$\exists Q.C \in CL_{\mathcal{F}}(C_0) \;\Rightarrow\; Q,\, Q^-,\, \neg Q,\, \neg Q^- \in CL_{\mathcal{F}}(C_0)$$

The cardinality of $CL_{\mathcal{F}}(C_0)$ is linear in the length of C_0.

It can be shown, following the lines of the proof in [122] for converse deterministic PDL, that $\mathcal{ALCFI}_{reg}$ enjoys the *tree-model property*, i.e., every satisfiable concept has a model that has the structure of a (possibly infinite) tree with branching degree linearly bounded by the size of the concept. More precisely, the following result holds.

[5]No condition is imposed on the finite paths of the run.

[6]We remind that C and C' stand for arbitrary concepts, and R and R' stand for arbitrary roles.

Theorem 5.1 *Every satisfiable $\mathcal{ALCFI}_{reg}$ concept C_0 has a tree model with branching degree k_{C_0} equal to twice the number of elements of $CL_{\mathcal{F}}(C_0)$.*

This property allows us to check satisfiability of an $\mathcal{ALCFI}_{reg}$ concept C_0 by building a 2ATA that accepts the (labeled) trees that correspond to tree models of C_0. Let $\mathcal{A}$ be the set of atomic concepts appearing in C_0, and $\mathcal{B}$ the set of atomic roles appearing in C_0 and their inverses. We construct from the $\mathcal{ALCFI}_{reg}$ concept C_0 a 2ATA $\mathbf{A}^{\mathcal{F}}_{C_0}$ that checks that C_0 is satisfied at the root of the input tree. We represent in each node of the tree the information about which atomic concepts are true in the node, and about the basic role that connects the predecessor of the node to the node itself (except for the root). More precisely, we label each node x with a set σ of atomic concepts and basic roles. The atomic concepts in σ are those that are true in x, and (for $\mathcal{ALCFI}_{reg}$) σ contains, except for the root, a single basic role, which is the one through which x is reached from its predecessor. That is, if Q stands for an atomic role P, then x is reached from its predecessor through P, and if Q stands for P^-, then the predecessor is reached from x through P. In the root, σ contains no basic role.

Given an $\mathcal{ALCFI}_{reg}$ concept C_0, we construct an automaton $\mathbf{A}^{\mathcal{F}}_{C_0}$ that accepts trees that correspond to tree models of C_0. For technical reasons, it is convenient to consider concepts in *negation normal form* (i.e., negations are pushed inside as much as possible). It is easy to check that the transformation of a concept into an equivalent one in negation normal form can be performed in linear time in the size of the concept. Below, we denote by $\mathit{nnf}(C)$ the negation normal form of C, and with $CL^{nnf}_{\mathcal{F}}(C_0)$ the set $\{\mathit{nnf}(C) \mid C \in CL_{\mathcal{F}}(C_0)\}$. The automaton $\mathbf{A}^{\mathcal{F}}_{C_0} = (\Sigma, S, \delta, s_{ini}, F)$ is defined as follows.

- The alphabet is $\Sigma = \bigcup_{Q \in \mathcal{B}} 2^{\mathcal{A} \cup \{Q\}}$, i.e., all sets consisting of atomic concepts and at most one basic role. This corresponds to labeling each node of the tree with a truth assignment to the atomic concepts, and with the role used to reach the node from its predecessor.

- The set of states is $S = \{s_{ini}\} \cup CL^{nnf}_{\mathcal{F}}(C_0)$, where s_{ini} is the initial state[7]. Intuitively, when the automaton is in a state $\sigma \in CL^{nnf}_{\mathcal{F}}(C_0)$ and visits a node x of the tree, this means that the automaton has to check that σ holds in x. When σ is an atomic concept A (resp. a basic role Q), this amounts to check that the node label contains A (resp. Q).

- The set F of final states is the set of concepts in $CL^{nnf}_{\mathcal{F}}(C_0)$ of the form $\forall R^*.C$. Observe that concepts

[7]Recall that $CL^{nnf}_{\mathcal{F}}(C_0)$ contains also the atomic roles appearing in C_0 and their inverses.

of the form $\exists R^*.C$ are not final states, and this is sufficient to guarantee that such concepts are satisfied in all accepting runs of the automaton.

- The transition function δ is defined below.

1. For each $\sigma \in 2^{\mathcal{A}}$, i.e., containing no basic role, there is a transition from the initial state

$$\delta(s_{ini}, \sigma) \;=\; (0, \mathit{nnf}(C_0))$$

Such a transition checks that the root of the tree is not labeled with any basic role, and moves to the state that verifies C_0 in the root itself.

2. For each $\sigma \in \Sigma$, and each atomic concept or basic role $s \in \mathcal{A} \cup \mathcal{B}$ there are transitions

$$\begin{array}{rcl} \delta(s,\sigma) & = & \begin{cases} \textbf{true} & \text{if } s \in \sigma \\ \textbf{false} & \text{if } s \notin \sigma \end{cases} \\ \delta(\neg s,\sigma) & = & \begin{cases} \textbf{true} & \text{if } s \notin \sigma \\ \textbf{false} & \text{if } s \in \sigma \end{cases} \end{array}$$

For $s \in \mathcal{A}$, such transitions check the truth value of atomic concepts and their negations in the current node of the tree. For $s \in \mathcal{B}$, such transitions check through which role the current node is reached.

3. For the concepts in $CL^{nnf}_{\mathcal{F}}(C_0)$ and each $\sigma \in \Sigma$ there are transitions

$$\begin{array}{rcl} \delta(C \sqcap C', \sigma) & = & (0, C) \;\wedge\; (0, C') \\ \delta(C \sqcup C', \sigma) & = & (0, C) \;\vee\; (0, C') \\ \delta(\forall Q.C, \sigma) & = & ((0, \neg Q^-) \vee (-1, C)) \wedge \\ & & \bigwedge_{1 \leq i \leq k_{C_0}} ((i, \neg Q) \vee (i, C)) \\ \delta(\forall(R \cup R').C, \sigma) & = & (0, \forall R.C) \;\wedge\; (0, \forall R'.C) \\ \delta(\forall(R \circ R').C, \sigma) & = & (0, \forall R.\forall R'.C) \\ \delta(\forall R^*.C, \sigma) & = & (0, C) \;\wedge\; (0, \forall R.\forall R^*.C) \\ \delta(\forall id(C).C', \sigma) & = & (0, \mathit{nnf}(\neg C)) \;\vee\; (0, C') \\ \delta(\exists Q.C, \sigma) & = & ((0, Q^-) \wedge (-1, C)) \vee \\ & & \bigvee_{1 \leq i \leq k_{C_0}} ((i, Q) \wedge (i, C)) \\ \delta(\exists(R \cup R').C, \sigma) & = & (0, \exists R.C) \;\vee\; (0, \exists R'.C) \\ \delta(\exists(R \circ R').C, \sigma) & = & (0, \exists R.\exists R'.C) \\ \delta(\exists R^*.C, \sigma) & = & (0, C) \;\vee\; (0, \exists R.\exists R^*.C) \\ \delta(\exists id(C).C', \sigma) & = & (0, C) \;\wedge\; (0, C') \end{array}$$

All such transitions, except for those involving $\forall R^*.C$ and $\exists R^*.C$, inductively decompose concepts and roles, and move to appropriate states of the automaton and nodes of the tree. The transitions involving $\forall R^*.C$ treat $\forall R^*.C$ as the equivalent concept $C \sqcap \forall R.\forall R^*.C$, and the transitions involving $\exists R^*.C$ treat $\exists R^*.C$ as the equivalent concept $C \sqcup \exists R.\exists R^*.C$.

4. For each concept of the form $(\leqslant 1\, Q)$ in $CL^{nnf}_{\mathcal{F}}(C)$ and each $\sigma \in \Sigma$ there is a transition

$$\begin{array}{l} \delta((\leqslant 1\, Q), \sigma) \;=\; \\ \quad ((0, Q^-) \;\wedge\; \bigwedge_{1 \leq i \leq k_{C_0}} (i, \neg Q)) \;\vee \\ \quad ((0, \neg Q^-) \;\wedge\; \bigwedge_{1 \leq i < j \leq k_{C_0}} ((i, \neg Q) \vee (j, \neg Q))) \end{array}$$

Such transitions check that, for a node x labeled with $(\leqslant 1\, Q)$, there exists at most one node (among the predecessor and the successors of x) reachable from x through Q.

5. For each concept of the form $\neg(\leqslant 1\, Q)$ in $CL^{nnf}_{\mathcal{F}}(C)$ and each $\sigma \in \Sigma$ there is a transition

$$\begin{array}{l} \delta(\neg(\leqslant 1\, Q), \sigma) \;=\; \\ \quad ((0, Q^-) \;\wedge\; \bigvee_{1 \leq i \leq k_{C_0}} (i, Q)) \;\vee \\ \quad \bigvee_{1 \leq i < j \leq k_{C_0}} ((i, Q) \wedge (j, Q)) \end{array}$$

Such transitions check that, for a node x labeled with $\neg(\leqslant 1\, Q)$, there exist at least two nodes (among the predecessor and the successors of x) reachable from x through Q.

A run of the automaton $\mathbf{A}^{\mathcal{F}}_{C_0}$ on an infinite tree starts in the root checking that C_0 holds there (item 1 above). It does so by inductively decomposing $\mathit{nnf}(C_0)$ while appropriately navigating the tree (item 3) until it arrives to atomic concepts, functional restrictions, and their negations. These are checked locally (items 2, 4 and 5). Concepts of the form $\forall R^*.C$ and $\exists R^*.C$ are propagated using the equivalent concepts $C \sqcap \forall R.\forall R^*.C$ and $C \sqcup \exists R.\exists R^*.C$, respectively. It is only the propagation of such concepts that may generate infinite branches in a run. Now, a run of the automaton may contain an infinite branch in which $\exists R^*.C$ is always resolved by choosing the disjunct $\exists R.\exists R^*.C$, without ever choosing the disjunct C. This infinite branch in the run corresponds to an infinite path in the tree where R is iterated forever and in which C is never fulfilled. However, the semantics of $\exists R^*.C$ requires that C is fulfilled after a finite number of iterations of R. Hence such an infinite path cannot be used to satisfy $\exists R^*.C$. The acceptance condition of the automaton, which requires that each infinite branch in a run contains a state of the form $\forall R^*.C$, rules out such infinite branches in accepting runs. Indeed, a run always deferring the fulfillment of C will contain an infinite branch where all states have the form $\exists R_1.\cdots\exists R_n.\exists R^*.C$, with $n \geq 0$ and $R_1 \circ \cdots \circ R_n$ a postfix of R. Observe that the only remaining infinite branches in a run are those that arise by propagating concepts of the form $\forall R^*.C$ indefinitely often. The acceptance condition allows for such branches.

Given a labeled tree $\mathcal{T} = (T, V)$ accepted by $\mathbf{A}^{\mathcal{F}}_{C_0}$, one can define an interpretation $\mathcal{I}_{\mathcal{T}}$ that is a model of C_0. Conversely, given a tree model $\mathcal{I}$ of C_0 with branching degree k_{C_0}, one can obtain a labeled tree $\mathcal{T}_{\mathcal{I}}$ (with branching degree k_{C_0}) that is accepted by $\mathbf{A}^{\mathcal{F}}_{C_0}$. Hence, an $\mathcal{ALCFI}_{reg}$ concept C_0 is satisfiable if and only if the set of trees accepted by $\mathbf{A}^{\mathcal{F}}_{C_0}$ is not empty. It follows that we can use algorithms for nonemptiness of 2ATAs to check satisfiability in $\mathcal{ALCFI}_{reg}$. It turns out that such a decision procedure is indeed optimal w.r.t. the computational complexity. The 2ATA $\mathbf{A}^{\mathcal{F}}_{C_0}$ has a number of states that is linear in the size of C_0, while the alphabet is exponential in the number of atomic concepts occurring in C_0. By the complexity characterization of non-emptiness of 2ATAs mentioned above [121], we get an upper bound for reasoning in $\mathcal{ALCFI}_{reg}$ that matches the EXPTIME lower bound.

Theorem 5.2 *Concept satisfiability (and hence logical implication) in $\mathcal{ALCFI}_{reg}$ is* EXPTIME*-complete.*

The construction above can be adapted to deal with several other construct. For example, by adding to the automaton suitable states and transitions that count the number of successors of a node that satisfy a certain concept, one can deal with qualified number restrictions [43, 87]. Similarly, one can easily extend the automaton to handle intersection, union, and difference of basic roles (i.e., atomic roles and their inverses) [117, 43]. By changing the acceptance condition of the automaton to a parity condition, one can deal with fixpoint constructs [121, 39, 107].

6 Conclusions

In the previous sections we provided an overview of the most important characteristics of DLs, focusing mainly on the problem of devising automated techniques for reasoning in these logics. We conclude the paper by mentioning several other important issues in the research on DLs, that have not been addressed here.

- There are other interesting modeling features that have been investigated in the context of DLs. Among them we mention: ABox [63], n-ary relations and data dependencies [53, 25, 85, 42], concrete domains [6, 93, 94], queries [89, 92, 17, 65, 37, 41, 18].

- Following the experience of early terminological systems like [96, 27, 103, 7], DL systems such as FACT [81] and RACER [78] have proved to be sufficiently efficient to be used in several interesting applications. Furthermore, several tools exploiting DL reasoners in specific domain applications have been built in recent years [72, 14].

- Besides classical first-order reasoning, other types of reasoning have been investigated for DLs, such as nonmonotonic and epistemic reasoning [8, 62], finite model reasoning [46, 34], unification [9], computation of the Least Common Subsumers [10], and rewriting of concept expressions [15, 11].

Finally, we did not discuss applications of DLs in this paper. The interested reader will find details on this subject and on all the above mentioned issues in the forthcoming Handbook on DLs [5].

References

[1] C. Areces, P. Blackburn, and M. Marx. Hybrid logic: Characterization, interpolation and complexity. *J. of Symbolic Logic*, 66(3):977–1010, 2001.

[2] A. Artale and E. Franconi. A computational account for a description logic of time and action. In *Proc. of KR'94*, pages 3–14, 1994.

[3] F. Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In *Proc. of IJCAI'91*, 1991.

[4] F. Baader, H.-J. Bürckert, B. Hollunder, W. Nutt, and J. H. Siekmann. Concept logics. In J. W. Lloyd, editor, *Computational Logics, Symposium Proceedings*, pages 177–201. Springer, 1990.

[5] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2002. To appear.

[6] F. Baader and P. Hanschke. A schema for integrating concrete domains into concept languages. In *Proc. of IJCAI'91*, pages 452–457, 1991.

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

[8] F. Baader and B. Hollunder. Embedding defaults into terminological knowledge representation formalisms. In *Proc. of KR'92*, pages 306–317. Morgan Kaufmann, Los Altos, 1992.

[9] F. Baader and R. Küsters. Unification in a description logic with transitive closure of roles. In R. Nieuwenhuis and A. Voronkov, editors, *Proc. of LPAR 2001*, volume 2250 of *LNCS*, pages 217–232. Springer, 2001.

[10] F. Baader, R. Küsters, and R. Molitor. Computing least common subsumers in description logics with existential restrictions. In *Proc. of IJCAI'99*, pages 96–101, 1999.

[11] F. Baader, R. Küsters, and R. Molitor. Rewriting concepts using terminologies. In *Proc. of KR 2000*, pages 297–308, 2000.

[12] F. Baader, C. Lutz, H. Sturm, and F. Wolter. Fusions of description logics. In *Proc. of DL 2000*, pages 21–30. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-33/`, 2000.

[13] F. Baader and U. Sattler. Tableau algorithms for description logics. In R. Dyckhoff, editor, *Proc. of TABLEAUX 2000*, volume 1847 of *LNAI*, pages 1–18. Springer, 2000.

[14] S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens. OilEd: A Reason-able ontology editor for the semantic web. In *Proc. of KI 2001*, number 2174 in LNAI, pages 396–408. Springer, 2001. Appeared also in Proc. of DL 2001.

[15] C. Beeri, A. Y. Levy, and M.-C. Rousset. Rewriting queries using views in description logics. In *Proc. of PODS'97*, pages 99–108, 1997.

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

[17] S. Bergamaschi, D. Beneventano, C. Sartori, and M. Vincini. ODB-QOPTIMIZER: A tool for semantic query optimization in OODB. In *Proc. of ICDE'97*, 1997.

[18] S. Bergamaschi, S. Castano, M. Vincini, and D. Beneventano. Semantic integration of heterogeneous information sources. *Data and Knowledge Engineering*, 36(3):215–249, 2001.

[19] S. Bergamaschi and C. Sartori. On taxonomic reasoning in conceptual design. *ACM Trans. on Database Systems*, 17(3):385–422, 1992.

[20] O. Bernholtz, M. Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. In *Proc. of CAV'94*, volume 818 of *LNCS*, pages 142–155. Springer, 1994.

[21] P. Blackburn. Nominal tense logic. *Notre Dame Journal of Formal Logic*, 34(1):56–83, 1993.

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

[23] J. L. Blanco, A. Illarramendi, and A. Goñi. Building a federated relational database system: An approach using a knowledge-based system. *J. of Intelligent and Cooperative Information Systems*, 3(4):415–455, 1994.

[24] A. Borgida. Description logics in data management. *IEEE Trans. on Knowledge and Data Engineering*, 7(5):671–682, 1995.

[25] A. Borgida and G. E. Weddell. Adding uniqueness constraints to description logics (preliminary report). In *Proc. of DOOD'97*, pages 85–102, 1997.

[26] R. J. Brachman and H. J. Levesque. The tractability of subsumption in frame-based description languages. In *Proc. of AAAI'84*, pages 34–37, 1984.

[27] R. J. Brachman, D. L. McGuinness, P. F. Patel-Schneider, L. A. Resnick, and A. Borgida. Living with CLASSIC: When and how to use a KL-ONE-like language. In J. F. Sowa, editor, *Principles of Semantic Networks*, pages 401–456. Morgan Kaufmann, Los Altos, 1991.

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

[29] T. Bray, J. Paoli, and C. M. Sperberg-McQueen. Extensible Markup Language (XML) 1.0 — W3C recommendation. Technical report, World Wide Web Consortium, 1998. Available at `http://www.w3.org/TR/1998/REC-xml-19980210`.

[30] M. Buchheit, F. M. Donini, W. Nutt, and A. Schaerf. Terminological systems revisited: Terminology = schema + views. In *Proc. of AAAI'94*, pages 199–204, 1994.

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

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

[33] P. Buneman. Semistructured data. In *Proc. of PODS'97*, pages 117–121, 1997.

[34] D. Calvanese. Finite model reasoning in description logics. In *Proc. of KR'96*, pages 292–303, 1996.

[35] D. Calvanese. *Unrestricted and Finite Model Reasoning in Class-Based Representation Formalisms*. PhD thesis, Dip. di Inf. e Sist., Univ. di Roma "La Sapienza", 1996. Available at `http://www.dis.uniroma1.it/pub/calvanes/thesis.ps.gz`.

[36] D. Calvanese, G. De Giacomo, and M. Lenzerini. Structured objects: Modeling and reasoning. In *Proc. of DOOD'95*, volume 1013 of *LNCS*, pages 229–246. Springer, 1995.

[37] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proc. of PODS'98*, pages 149–158, 1998.

[38] D. Calvanese, G. De Giacomo, and M. Lenzerini. What can knowledge representation do for semi-structured data? In *Proc. of AAAI'98*, pages 205–210, 1998.

[39] D. Calvanese, G. De Giacomo, and M. Lenzerini. Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In *Proc. of IJCAI'99*, pages 84–89, 1999.

[40] D. Calvanese, G. De Giacomo, and M. Lenzerini. Representing and reasoning on XML documents: A description logic approach. *J. of Log. and Comp.*, 9(3):295–318, 1999.

[41] D. Calvanese, G. De Giacomo, and M. Lenzerini. Answering queries using views over description logics knowledge bases. In *Proc. of AAAI 2000*, pages 386–391, 2000.

[42] D. Calvanese, G. De Giacomo, and M. Lenzerini. Identification constraints and functional dependencies in description logics. In *Proc. of IJCAI 2001*, pages 155–160, 2001.

[43] D. Calvanese, G. De Giacomo, and M. Lenzerini. 2ATAs make DLs easy. In *Proc. of DL 2002*, pages 107–118. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-53/`, 2002.

[44] D. Calvanese, G. De Giacomo, M. Lenzerini, and D. Nardi. Reasoning in expressive description logics. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, chapter 23, pages 1581–1634. Elsevier, 2001.

[45] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Description logic framework for information integration. In *Proc. of KR'98*, pages 2–13, 1998.

[46] D. Calvanese and M. Lenzerini. Making object-oriented schemas more expressive. In *Proc. of PODS'94*, pages 243–254, 1994.

[47] D. Calvanese, M. Lenzerini, and D. Nardi. A unified framework for class based representation formalisms. In *Proc. of KR'94*, pages 109–120, 1994.

[48] D. Calvanese, M. Lenzerini, and D. Nardi. Unifying class-based representation formalisms. *J. of Artificial Intelligence Research*, 11:199–240, 1999.

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

[50] G. De Giacomo. *Decidability of Class-Based Knowledge Representation Formalisms*. PhD thesis, Dip. di Inf. e Sist., Univ. di Roma "La Sapienza", 1995.

[51] G. De Giacomo and M. Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proc. of AAAI'94*, pages 205–212, 1994.

[52] G. De Giacomo and M. Lenzerini. Concept language with number restrictions and fixpoints, and its relationship with μ-calculus. In *Proc. of ECAI'94*, pages 411–415, 1994.

[53] G. De Giacomo and M. Lenzerini. Description logics with inverse roles, functional restrictions, and n-ary relations. In *Proc. of JELIA'94*, volume 838 of *LNAI*, pages 332–346. Springer, 1994.

[54] G. De Giacomo and M. Lenzerini. What's in an aggregate: Foundations for description logics with tuples and sets. In *Proc. of IJCAI'95*, pages 801–807, 1995.

[55] G. De Giacomo and M. Lenzerini. TBox and ABox reasoning in expressive description logics. In *Proc. of KR'96*, pages 316–327, 1996.

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

[57] G. De Giacomo and F. Massacci. Combining deduction and model checking into tableaux and algorithms for converse-PDL. *Information and Computation*, 160(1–2):117–137, 2000.

[58] P. Devanbu and M. A. Jones. The use of description logics in KBSE systems. *ACM Trans. on Software Engineering and Methodology*, 6(2):141–172, 1997.

[59] F. M. Donini, B. Hollunder, M. Lenzerini, A. M. Spaccamela, D. Nardi, and W. Nutt. The complexity of existential quantification in concept languages. *Artificial Intelligence*, 2–3:309–327, 1992.

[60] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. Tractable concept languages. In *Proc. of IJCAI'91*, pages 458–463, 1991.

[61] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. *Information and Computation*, 134:1–58, 1997.

[62] F. M. Donini, M. Lenzerini, D. Nardi, W. Nutt, and A. Schaerf. An epistemic operator for description logics. *Artificial Intelligence*, 100(1–2):225–274, 1998.

[63] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Deduction in concept languages: From subsumption to instance checking. *J. of Log. and Comp.*, 4(4):423–452, 1994.

[64] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in description logics. In G. Brewka, editor, *Principles of Knowledge Representation*, Studies in Logic, Language and Information, pages 193–238. CSLI Publications, 1996.

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

[66] F. M. Donini and F. Massacci. EXPTIME tableaux for $\mathcal{ALC}$. *Artificial Intelligence*, 124(1):87–138, 2000.

[67] E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy. In *Proc. of FOCS'91*, pages 368–377, 1991.

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

[69] D. Fensel, F. van Harmelen, I. Horrocks, D. L. McGuinness, and P. F. Patel-Schneider. OIL: An ontology infrastructure for the semantic web. *IEEE Intelligent Systems*, 16(2):38–45, 2001.

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

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

[72] E. Franconi and G. Ng. The i.com tool for intelligent conceptual modeling. In *Proc. of KRDB 2000*, pages 45–53. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-29/`, 2000.

[73] G. Gargov and V. Goranko. Modal logic with names. *J. of Philosophical Logic*, 22:607–636, 1993.

[74] G. Gargov and S. Passy. Determinism and looping in combinatory PDL. *Theor. Comp. Sci.*, 61:259–277, 1988.

[75] F. Goasdoue, V. Lattes, and M.-C. Rousset. The use of CARIN language and algorithms for information integration: The Picsel system. *Int. J. of Cooperative Information Systems*, 9(4):383–401, 2000.

[76] E. Gonçalvès and E. Grädel. Decidability issues for action guarded logics. In *Proc. of DL 2000*, pages 123–132. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-33/`, 2000.

[77] V. Haarslev and R. Möller. High performance reasoning with very large knowledge bases: A practical case study. In *Proc. of IJCAI 2001*, pages 161–168, 2001.

[78] V. Haarslev and R. Möller. RACER system description. In *Proc. of IJCAR 2001*, volume 2083 of *LNAI*, pages 701–705. Springer, 2001.

[79] B. Hollunder and F. Baader. Qualifying number restrictions in concept languages. Technical Report RR-91-03, DFKI, Kaiserslautern (Germany), 1991. An abridged version appeared in *Proc. of KR'91*.

[80] B. Hollunder and F. Baader. Qualifying number restrictions in concept languages. In *Proc. of KR'91*, pages 335–346, 1991.

[81] I. Horrocks. Using an expressive description logic: FaCT or fiction? In *Proc. of KR'98*, pages 636–647, 1998.

[82] I. Horrocks. Daml+oil: a description logic for the semantic web. *IEEE Bull. on Data Engineering*, 25(1):4–9, 2002.

[83] I. Horrocks and P. F. Patel-Schneider. Optimizing description logic subsumption. *J. of Log. and Comp.*, 9(3):267–293, 1999.

[84] I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *J. of Log. and Comp.*, 9(3):385–410, 1999.

[85] V. L. Khizder, D. Toman, and G. E. Weddell. On decidability and complexity of description logics with uniqueness constraints. In *Proc. of ICDT 2001*, 2001.

[86] D. Kozen and J. Tiuryn. Logics of programs. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science — Formal Models and Semantics*, pages 789–840. Elsevier, 1990.

[87] O. Kupferman and M. Y. Vardi. The complexity of the graded mu-calculus. In *Proc. of CADE 2002*, 2002.

[88] N. Kurtonina and M. de Rijke. Expressiveness of concept expressions in first-order description. *Artificial Intelligence*, 107(2):303–333, 1999.

[89] M. Lenzerini and A. Schaerf. Concept languages as query languages. In *Proc. of AAAI'91*, pages 471–476, 1991.

[90] H. J. Levesque and R. J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3:78–93, 1987.

[91] A. Y. Levy, A. Rajaraman, and J. J. Ordille. Query answering algorithms for information agents. In *Proc. of AAAI'96*, pages 40–47, 1996.

[92] A. Y. Levy and M.-C. Rousset. Combining Horn rules and description logics in CARIN. *Artificial Intelligence*, 104(1–2):165–209, 1998.

[93] C. Lutz. Reasoning with concrete domains. In *Proc. of IJCAI'99*, pages 90–95, 1999.

[94] C. Lutz. NEXPTIME-complete description logics with concrete domains. In *Proc. of IJCAR 2001*, volume 2083 of *LNAI*, pages 45–60. Springer, 2001.

[95] D. E. Muller and P. E. Schupp. Alternating automata on infinite trees. *Theor. Comp. Sci.*, 54:267–276, 1987.

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

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

[98] B. Nebel. Terminological cycles: Semantics and computational properties. In J. F. Sowa, editor, *Principles of Semantic Networks*, pages 331–361. Morgan Kaufmann, Los Altos, 1991.

[99] S. Passy and T. Tinchev. PDL with data constraints. *Information Processing Letters*, 20:35–41, 1985.

[100] S. Passy and T. Tinchev. An essay in combinatory dynamic logic. *Information and Computation*, 93:263–332, 1991.

[101] P. F. Patel-Schneider. Small can be beautiful in knowledge representation. In *Proc. of the IEEE Workshop on Knowledge-Based Systems*, 1984. An extended version appeared as Fairchild Tech. Rep. 660 and FLAIR Tech. Rep. 37, October 1984.

[102] P. F. Patel-Schneider, R. J. Brachman, and H. J. Levesque. ARGON: Knowledge representation meets information retrieval. In *Proc. of the 1st Conf. on Artificial Inteligence Applications*, 1984. Also available as Fairchild Technical Report 654 and FLAIR Technical Report 29.

[103] C. Peltason. The BACK system — an overview. *SIGART Bulletin*, 2(3):114–119, 1991.

[104] V. R. Pratt. A near-optimal method for reasoning about action. *J. of Computer and System Sciences*, 20:231–255, 1980.

[105] A. Prior. *Past, Present, and Future*. Oxford University Press, 1967.

[106] J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language Reference Manual*. Addison Wesley Publ. Co., Reading, Massachussetts, 1998.

[107] U. Sattler and M. Y. Vardi. The hybrid μ-calculus. In *Proc. of IJCAR 2001*, pages 76–91, 2001.

[108] A. Schaerf. On the complexity of the instance checking problem in concept languages with existential quantification. *J. of Intelligent Information Systems*, 2:265–278, 1993.

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

[110] K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of IJCAI'91*, pages 466–471, 1991.

[111] K. Schild. Terminological cycles and the propositional μ-calculus. In *Proc. of KR'94*, pages 509–520, 1994.

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

[113] A. P. Sheth, S. K. Gala, and S. B. Navathe. On automatic reasoning for schema integration. *J. of Intelligent and Cooperative Information Systems*, 2(1):23–50, 1993.

[114] W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 4, pages 133–192. Elsevier, 1990.

[115] W. Thomas. Languages, automata, and logic. In *Handbook of Formal Language Theory*, volume III, pages 389–455. 1997.

[116] S. Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *J. of Artificial Intelligence Research*, 12:199–217, 2000.

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

[118] J. D. Ullman. Information integration using logical views. In *Proc. of ICDT'97*, volume 1186 of *LNCS*, pages 19–40. Springer, 1997.

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

[120] W. Van der Hoek and M. de Rijke. Counting objects. *J. of Log. and Comp.*, 5(3):325–345, 1995.

[121] M. Y. Vardi. Reasoning about the past with two-way automata. In *Proc. of ICALP'98*, volume 1443 of *LNCS*, pages 628–641. Springer, 1998.

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

[123] R. Weida and D. Litman. Terminological reasoning with constraint networks and an application to plan recognition. In *Proc. of KR'92*, pages 282–293, 1992.

[124] W. A. Woods and J. G. Schmolze. The KL-ONE family. In F. W. Lehmann, editor, *Semantic Networks in Artificial Intelligence*, pages 133–178. Pergamon Press, 1992. Published as a special issue of *Computers & Mathematics with Applications*, Volume 23, Number 2–9.