Reasoning and Explanation in $\mathcal{EL}$ and in Expressive Description Logics

Anni-Yasmin Turhan

Theoretical Computer Science,
TU Dresden, Germany,
turhan@tcs.inf.tu-dresden.de

Abstract. Description Logics (DLs) are the formalism underlying the standard web ontology language OWL 2. DLs have formal semantics which are the basis for powerful reasoning services. In this paper, we introduce the basic notions of DLs and the techniques that realize subsumption—the fundamental reasoning service of DL systems. We discuss two reasoning methods for this service: the tableau method for expressive DLs such as $\mathcal{ALC}$ and the completion method for the light-weight DL $\mathcal{EL}$. We also present methods for generating explanations for computed subsumption relationships in these two DLs.

1 Introduction

The ontology language for the semantic web OWL provides means to describe entities of a application domain in an ontology. The underlying formalism for OWL are Description Logics, which have well-defined syntax and formal semantics. The recent version of the W3C standard OWL 2.0 has four language variants: the OWL 2 language itself and three profiles. The latter are light-weight ontology languages of relatively low expressivity and that are tailored to be efficient for specific reasoning tasks. We are interested in the reasoning task of computing subsumption, i.e., sub- and super-class relationships, and providing explanations for the obtained reasoning results. In this paper, we discuss reasoning techniques for computing subsumption relationships for the core description logics underlying the OWL 2 language: $\mathcal{ALC}$ and the core description logics underlying the EL profile: $\mathcal{EL}$. The EL profile is particularly suitable for applications with ontologies that define very large numbers of classes and that need subsumption as the main inference service. Based on the reasoning techniques for subsumption, we discuss methods to compute explanations for detected subsumption relationships in $\mathcal{ALC}$ and $\mathcal{EL}$. Before we turn to the reasoning techniques, we give general overview of Description Logics.

Description Logics (DLs) [6] are a family of knowledge representation formalisms that have formal semantics. This family of logics is tailored towards representing terminological knowledge of an application domain in a structured and formally well-understood way. Description logics allow users to define important notions, such as classes or relations of their application domain in terms of concepts and roles. These concepts (unary predicates) and roles (binary predicates) then restrict the way these classes and relations are interpreted. Based on these definitions, implicitly captured

knowledge can be inferred from the given descriptions of concepts and roles, as for instance sub-class or instance relationships.

The name *Description Logics* is motivated by the fact that classes and relations are defined in terms of concept *descriptions*. These concept descriptions are complex expressions built from atomic concepts and atomic roles using the concept constructors offered by the particular DL in use. Based on their formal semantics, a whole collection of inference services has been defined and investigated for different DLs. DLs have been employed in various domains, such as databases, biomedical or context-aware applications [3, 96]. Their most notable success so far is probably the adoption of the DL-based language OWL[1] as standard ontology language for the Semantic Web [53].

Historically, DLs stem from knowledge representation systems such as *semantic networks* [85, 94] or *frame systems* [73]. These early knowledge representation systems were motivated by linguistic applications and allow to specify information from the domain of discourse. They offer methods to compute inheritance relations between the specified notions. Early frame-based systems and semantic networks both have operational semantics, i.e., the semantics of reasoning is given by its implementation. As a consequence, the result of the reasoning process depends on the implementation of the reasoner and thus the result may differ from system to system for the same input [95]. To remedy this, DLs and their reasoning services are based on formal semantics. The information about the application domain is represented in a declarative and unambiguous way. More importantly, the formal semantics of the reasoning services ensure predictable and thus reliable behavior of the DL reasoning systems—independent of the implementation.

The investigation of algorithms for reasoning services and their complexity is the main focus of the DL research community. Typically, one can distinguish the following phases of DL research during the last decades. In the late eighties, reasoning algorithms have been devised for DL systems that mostly were sound, but incomplete, i.e., they would return correct answers, but would not find *all* correct answers. This development was led by the belief that terminological reasoning is inherently intractable [79, 80], and thus completeness was traded for tractability. These algorithms have been implemented in systems such as Classic [23, 22, 84] and Back [79, 81]. During the nineties, sound and complete reasoning methods were investigated for the core inferences of DL systems: consistency and subsumption. *Consistency* assures that the specification of the concepts, roles and individuals are free of contradictions. For *subsumption* one computes super- and sub-concept relations from the given specifications of concepts and roles. The use of incomplete algorithms for these inferences has largely been abandoned in the DL community since then, mainly because of the problem that the behavior of the systems is no longer determined by the semantics of the description language: an incomplete algorithm may claim that a subsumption relationship does not hold, although it should hold according to the semantics.

The underlying technique for computing the basic DL inferences is the tableau method [37], which was adapted to DLs in [91]. This method was extended to more and more expressive DLs (for an overview, see [17]). The gain in expressiveness came at the cost of higher complexity for the reasoning procedures—reasoning for the DLs in-

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

vestigated is PSpace-complete or even ExpTime-complete [66, 54, 98] (for an overview see [17, 31]).

Despite the high complexity, highly optimized DL reasoning systems were implemented based on the tableau method—most prominently the FaCT system [49] and RACER [43]. These systems employed optimization methods developed for DL reasoning based on tableaux [7, 48, 58, 45] and demonstrated that the high worst case complexities would hardly be encountered in practice [49, 52, 58, 42, 50, 100]. In fact, it turned out that these highly optimized implementations of the reasoning methods do perform surprisingly well on DL knowledge bases from practical applications.

Encouraged by these findings and driven by application needs researchers investigated tableau algorithms for even more expressive DLs [55, 56, 51, 57] in the last decade. At the same time, the idea of the Semantic Web emerged and DLs became the basis for the W3C standardized web ontology language OWL [53, 44]. This brought DLs into the attention of new users from various application areas, which in turn necessitated automated support of ontology services and motivated research on various new inferences for DLs. For instance,

- the generation of *explanations* of consequences that the DL reasoner detected [90, 83, 63, 61, 15],
- support for building ontologies by computing *generalizations* [10, 27, 18, 101, 35],
- *conjunctive queries* as a means to access the instance data of an ontology [76, 29, 30, 39, 82, 36, 67], and
- computing *modularizations* of an ontology as means to facilitate their reuse [38, 69, 33, 32, 70].

All of them are currently investigated reasoning services for DLs and most of them are implemented in specialized reasoners. At the same time, the need for faster reasoners for the afore mentioned basic inferences for DLs led to two developments. On the one hand, the new tableau-based reasoners for expressive DL were developed such as PELLET [93], FaCT++ [99, 100] and RACERPRO [86] and new reasoning methods for expressive DLs were investigated and implemented such as resolution [74, 76] in KAON2 and hyper-tableau [77, 78] in HERMIT. On the other hand, *light-weight DLs*, which are DLs with relatively limited expressivity, but good computational properties for specific reasoning tasks were designed [13]. Reasoning even for large ontologies written in these DLs can be done efficiently, since the respective reasoning methods are tractable. There are two "families" of lightweight DLs: the $\mathcal{EL}$ family [25, 4, 5], for which the subsumption and the instance problem are polynomial, and the DL Lite family [28, 30], for which the instance problem and query answering are polynomial. A member of each of these families is the DL corresponding to one of the profiles of the OWL 2 standard.

In this paper, we examine the basic reasoning services for DLs for the light-weight DL $\mathcal{EL}$ and for expressive DLs. In the next section, we give basic definitions for the fundamental DLs $\mathcal{ALC}$ and $\mathcal{EL}$. We introduce basic notions such as concept descriptions, TBoxes and ABoxes and their semantics. Based on this, we define the central reasoning services common to most DL systems. In Section 3, we discuss the reasoning methods for basic reasoning problems: we describe the tableau method for $\mathcal{ALC}$ and the

completion-based approach for $\mathcal{EL}$. In Section 4, we turn to another reasoning service, namely the computation of explanations for (probably unexpected) reasoning results. Again, we consider methods for expressive DLs and for $\mathcal{EL}$ for this task.

2 Basic Definitions

The central notion for DLs are *concept descriptions*, which can be built from concept names and so-called *concept constructors*. For instance, one can describe a course as an event given by a lecturer in the following way by a concept description:

$$\mathsf{Event} \sqcap \exists\, \mathsf{given\text{-}by}.\mathsf{Lecturer} \sqcap \exists\, \mathsf{has\text{-}topic}.\top$$

This concept description is a conjunction (indicated by $\sqcap$) of the concept Event, the existential restriction $\exists$ given-by.Lecturer and the existential restriction $\exists$ has-topic.$\top$. The first existential restriction consists of the role name given-by and concept Lecturer, which relates the Lecturer to the course. The latter existential restriction states that there is a topic (which is not specified).

In general, concept descriptions are built from the set of concept names N_C and the set of role names N_R using concept constructors. Every DL offers a different set of concept constructors. The DL $\mathcal{EL}$ allows only for the concept constructors that were used in the example concept description above.

Definition 1 ($\mathcal{EL}$-concept descriptions). *Let N_C be a set of concept names and N_R a set of role names. The set of $\mathcal{EL}$-concept descriptions is the smallest set such that*

- *all concept names are $\mathcal{EL}$-concept descriptions;*
- *if C and D are $\mathcal{EL}$-concept descriptions, then $C \sqcap D$ is also an $\mathcal{EL}$-concept description;*
- *if C is an $\mathcal{EL}$-concept description and $r \in N_R$, then $\exists r.C$ is also an $\mathcal{EL}$-concept description.*

If this set of concept constructors is extended to all Boolean connectors, i.e., extended by disjunction ($\sqcup$) and full negation ($\neg$), one obtains the DL $\mathcal{ALC}$. We can define $\mathcal{ALC}$-concept descriptions inductively.

Definition 2 ($\mathcal{ALC}$-concept descriptions). *Let N_C be a set of concept names and N_R a set of role names. The set of $\mathcal{ALC}$-concept descriptions is the smallest set such that*

- *all concept names are $\mathcal{ALC}$-concept descriptions;*
- *if C and D are $\mathcal{ALC}$-concept descriptions, then $\neg C$, $C \sqcap D$ and $C \sqcup D$ are also $\mathcal{ALC}$-concept descriptions;*
- *if C is an $\mathcal{ALC}$-concept description and $r \in N_R$, then $\exists r.C$ and $\forall r.C$ are also $\mathcal{ALC}$-concept descriptions.*

We call concept descriptions of the form $\exists r.C$ existential restrictions and concept descriptions of the form $\forall r.C$ value restrictions. The semantics of DL concept descriptions is given by means of interpretations.

Definition 3 (Semantics of $\mathcal{ALC}$-concept descriptions). *Let C and D be $\mathcal{ALC}$-concept descriptions and r a role name. An interpretation is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where the* domain *$\Delta^{\mathcal{I}}$ is a non-empty set and $\cdot^{\mathcal{I}}$ is a function that assigns to every concept name A a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and to every role name r a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. This function is extended to complex $\mathcal{ALC}$-concept descriptions as follows:*

- $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$;
- $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$;
- $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$;
- $(\exists r.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid$ *there is a* $y \in \Delta^{\mathcal{I}}$ *with* $(x, y) \in r^{\mathcal{I}}$ *and* $y \in C^{\mathcal{I}}\}$; *and*
- $(\forall r.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid$ *for all* $y \in \Delta^{\mathcal{I}}$, $(x, y) \in r^{\mathcal{I}}$ *implies* $y \in C^{\mathcal{I}}\}$.

This definitions clearly also captures the semantics of the less expressive DL $\mathcal{EL}$. Both, $\mathcal{EL}$ and $\mathcal{ALC}$ also offer the *top-concept* $\top$, which is always interpreted as the whole domain $\Delta^{\mathcal{I}}$. In addition $\mathcal{ALC}$ also offers the *bottom concept* $\bot$, which is always interpreted as the empty set. Now, with the $\mathcal{ALC}$-concept constructors at hand, one can, for instance, characterize a graduate CS student by the following concept description:

$$\exists \text{ studies-subject. CS} \sqcap (\text{Master-Student} \sqcup \text{PhD-Student})$$

Concept description like these are the main building blocks to model terminological knowledge.

2.1 Terminological Knowledge

A name can be assigned to a concept description by a *concept definition*. For instance, we can write $\text{Course} \equiv \text{Event} \sqcap \exists \text{ given-by.Lecturer} \sqcap \exists \text{ has-topic.} \top$ to supply a concept definition for the concept Course.

Definition 4 (Concept definition, general concept inclusion). *Let A be a concept name and C, D be (possibly) complex concept description.*

- *A* concept definition *is a statement of the form $A \equiv C$.*
- *A* general concept inclusion *(GCI for short) is a statement of the form $C \sqsubseteq D$.*

It is easy to see that every concept definition $A \equiv C$ can be expressed by two GCIs: $A \sqsubseteq C$ and $C \sqsubseteq A$. The terminological information expressed by GCIs is collected in the so-called TBox.

Definition 5 (TBox). *A finite set of GCIs is called a* TBox.

An interpretation is a model of a TBox $\mathcal{T}$, if it satisfies all GCIs, i.e., if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all $C \sqsubseteq D$ in $\mathcal{T}$.

If all concept descriptions in a TBox $\mathcal{T}$ are from a description logic $\mathcal{L}$, then we call $\mathcal{T}$ a $\mathcal{L}$-TBox.

If a concept definition $A \equiv C$ in a TBox uses a concept name B directly, i.e., B appears in C, or if B is used indirectly by the definitions of the names appearing in C, we say that the TBox is *cyclic*. Otherwise a TBox is *acyclic*.

Definition 6 (Unfoldable TBox). *A TBox $\mathcal{T}$ is a finite set of concept definitions that is acyclic and such that every concept name appears at most once on the left-hand side of the concept definitions in $\mathcal{T}$. Given a TBox $\mathcal{T}$, we call the concept name A a* defined concept*, if A occurs on the left-hand side of a concept definition in $\mathcal{T}$. All other concepts are called* primitive concepts*.*

One of the basic reasoning services in DL systems is to test for the *satisfiability* of a concept or a TBox, i.e., to test whether the information specified in it contains logical contradictions or not. In case the TBox contains a contradiction, any consequence can follow logically from the TBox. Moreover, if a TBox is not satisfiable, the specified information can hardly capture the intended meaning from an application domain. To test for satisfiability is often a first step for a user to check whether a TBox models something "meaningful".

Definition 7 (Concept satisfiability, TBox satisfiability). *Let C be a concept description and $\mathcal{T}$ a TBox. The concept description C is* satisfiable *iff it has a model, i.e., iff there exists an interpretation $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \emptyset$. A TBox $\mathcal{T}$ is* satisfiable *iff it has a model, i.e., an interpretation that satisfies all GCIs in $\mathcal{T}$.*

If a concept or TBox is not satisfiable, it is called *unsatisfiable*. Other typical reasoning services offered in DL systems test for equivalence or inclusion relations between concepts. In the latter case, if one concept of the TBox models a more general category than another one, we say that this concept *subsumes* the other one.

Definition 8 (Concept subsumption, concept equivalence). *Let C, D be two concept descriptions and $\mathcal{T}$ a (possibly empty) TBox. The concept description C is* subsumed *by the concept description D w.r.t. $\mathcal{T}$ ($C \sqsubseteq_{\mathcal{T}} D$), iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds in every model $\mathcal{I}$ of $\mathcal{T}$. Two concepts C, D are* equivalent w.r.t. $\mathcal{T}$ *($C \equiv_{\mathcal{T}} D$), iff $C^{\mathcal{I}} = D^{\mathcal{I}}$ holds for every model $\mathcal{I}$ of $\mathcal{T}$.*

The computation of the subsumption relations for all named concepts mentioned in the TBox $\mathcal{T}$ is called *classification* of the TBox $\mathcal{T}$ and yields the *concept hierarchy* of the TBox $\mathcal{T}$.

2.2 Assertional Knowledge

Facts about individuals from the application domain can be stated by *assertions*. There are two basic kinds of assertions for DL systems—one expresses that an individual belongs to a concept and the other one specifies that two individuals are related via a role. The set N_I is the set of all individual names.

Definition 9 (Assertion, ABox). *Let C be a concept description, $r \in NR$ a role name and i, j ($\{i, j\} \subseteq N_I$) be two individual names, then*

- *$C(i)$ is called a* concept assertion *and*
- *$r(i, j)$ is called a* role assertion*.*

An ABox $\mathcal{A}$ is a finite set of concept assertions and role assertions.

For instance, we can express that Dresden is a city located at the river Elbe by the following ABox:

$$\{\ \mathsf{City(Dresden)},\ \mathsf{River(Elbe)},\ \mathsf{located\text{-}at(Dresden,\ Elbe)}\ \}$$

If all concept descriptions in an ABox $\mathcal{A}$ are from a Description Logic $\mathcal{L}$, then we call $\mathcal{A}$ a $\mathcal{L}$-ABox. In order to capture ABoxes, the interpretation function is now extended to individual names. Each individual name is mapped by the interpretation function to an element of the domain $\Delta^{\mathcal{I}}$.

Definition 10 (Semantics of assertions, semantics of ABoxes). *Let C be a concept description, r a role name and i, j two individual names, then an interpretation $\mathcal{I}$ satisfies*

- *the concept assertion $C(i)$ if $i^{\mathcal{I}} \in C^{\mathcal{I}}$ and*
- *the role assertion $r(i, j)$ if $(i^{\mathcal{I}}, j^{\mathcal{I}}) \in r^{\mathcal{I}}$.*

An interpretation $\mathcal{I}$ is a model of an ABox $\mathcal{A}$, if $\mathcal{I}$ satisfies every assertion in $\mathcal{A}$.

A DL *knowledge base* $\mathcal{K}$ consists of an ABox $\mathcal{A}$ and a TBox $\mathcal{T}$. We write $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. We can now test for the absence of contradictions in ABoxes.

Definition 11 (ABox consistency, instance of). *An ABox $\mathcal{A}$ is* consistent *w.r.t. a TBox $\mathcal{T}$, iff it has a model that is also a model for $\mathcal{T}$. The individual i is an* instance of *the concept description C w.r.t. an ABox $\mathcal{A}$ and a TBox $\mathcal{T}$ (we write $\mathcal{A} \models_{\mathcal{T}} C(i)$), iff $i^{\mathcal{I}} \in C^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{A}$.*

ABox realization is a reasoning service that computes for each individual i of an ABox $\mathcal{A}$ and a TBox $\mathcal{T}$ the set of all named concepts A appearing in $\mathcal{A}$ and $\mathcal{T}$ that (1) have i as an instance ($\mathcal{A} \models_{\mathcal{T}} A(i)$) and (2) that is least w.r.t. $\sqsubseteq_{\mathcal{T}}$.

Typically, all the reasoning services introduced in this section are implemented in DL systems. In Section 3, we discuss the reasoning algorithms for these inferences for $\mathcal{ALC}$ and in more detail for $\mathcal{EL}$. Before we do so, we survey some extensions of these two basic DLs.

2.3 Extensions of Basic DLs

The basic DL $\mathcal{ALC}$ has been extended in many ways and, as mentioned in the introduction, reasoning algorithms have been devised for many of these extensions, see [31]. We consider here now some of those extensions that are captured in the OWL 2 standard [102] and that are also covered in the OWL 2 EL profile [75]. The DLs underlying these standardized ontology languages are $\mathcal{SROIQ}$ [51] and $\mathcal{EL}^{++}$ [5], respectively. Both DLs allow to specify more information on roles.

A role r can be declared to be a *transitive role* in the TBox. The semantics is straight-forward. An interpretation $\mathcal{I}$ satisfies a transitive role declaration $\mathsf{transitive}(r)$ if $\{(a, b), (b, c)\} \subseteq r^{\mathcal{I}}$ implies $(a, c) \in r^{\mathcal{I}}$. Transitive roles can be used in concept descriptions. Assume that the role has-part is transitive, then the two axioms:

$$\mathsf{Summer\text{-}school} \equiv \exists\ \mathsf{has\text{-}part.\ Course}$$
$$\mathsf{Course} \equiv \exists\ \mathsf{has\text{-}part.\ Lesson}$$

imply that a Summer school has a part that is a lesson. The declaration of an *inverse role* applies to a role name r and yields its inverse r^{-1}, where the semantics is the obvious one, i.e.,

$$(r^{-1})^{\mathcal{I}} := \{(e, d) \mid (d, e) \in r^{\mathcal{I}}\}.$$

Using the inverse of the role attends, we can define the concept of a speaker giving a boring talk as

$$\mathsf{Speaker} \sqcap \exists \mathsf{gives}.(\mathsf{Talk} \sqcap \forall \mathsf{attends}^{-1}.(\mathsf{Bored} \sqcup \mathsf{Sleeping})).$$

Furthermore, it can be specified that a role is a super-role of another role by a *role inclusion axiom*. The set of all role inclusions form the *role hierarchy*. An interpretation $\mathcal{I}$ satisfies a role inclusion axiom $r \sqsubseteq s$ if $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$.

For instance, we might capture the fact that everybody who is attending something (a course) is also interested in this (course) by a role inclusion axiom

$$\mathsf{attends} \sqsubseteq \mathsf{interested\text{-}in}.$$

DL researchers have introduced many additional constructors to the basic DL $\mathcal{ALC}$ and investigated various DLs obtained by combining such constructors. Here, we only introduce qualified number restrictions as example for additional concept constructors. This extension is covered also in the DL $\mathcal{SROIQ}$, but not in $\mathcal{EL}^{++}$. See [1] for an extensive list of additional concept and role constructors.

Qualified number restrictions are of the form $(\geq n\, r.C)$ (at-least restriction) and $(\leq n\, r.C)$ (at-most restriction), where $n \geq 0$ is a non-negative integer, $r \in N_R$ is a role name, and C is a concept description. The semantics of these additional constructors is defined as follows:

$$(\geq n\, r.C)^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid \mathit{card}(\{e \mid (d, e) \in r^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\}) \geq n\},$$
$$(\leq n\, r.C)^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid \mathit{card}(\{e \mid (d, e) \in r^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\}) \leq n\},$$

where $\mathit{card}(X)$ yields the cardinality of the set X. Using qualified number restrictions, we can define the concept of all persons that attend at most 20 talks, of which at least 3 have the topic DL:

$$\mathsf{Person} \sqcap (\leq 20\, \mathsf{attends.Talk}) \sqcap (\geq 3\, \mathsf{attends.}(\mathsf{Talk} \sqcap \exists \mathsf{topic.DL})).$$

2.4 Relations of DLs to Other Logics

Description logics are logic-based knowledge representation formalisms. A natural question is how they are related to other logics. In fact, it is easy to see, given their semantics, that most description logics are a fragment of first order logic (FOL). Concept descriptions can be translated into FOL formulae with one free variable. Concept names can be interpreted as unary predicates and role names as binary relations, see for example [88, 68, 59]. An arbitrary $\mathcal{ALC}$-concept description can be translated into a FOL formula τ_x, where x is a free variable in the following way:

- $\tau_x(A) := A(x)$ for a concept name A,

- $\tau_x(\neg C) := \neg\, \tau_x(C)$,
- $\tau_x(C \sqcap D) := \tau_x(C) \wedge \tau_x(D)$,
- $\tau_x(C \sqcup D) := \tau_x(C) \vee \tau_x(D)$,
- $\tau_x(\exists r.C) := \exists y.(r(x,y) \vee \tau_y(C))$, where y is a variable different from x, and
- $\tau_x(\forall r.C) := \forall y.(r(x,y) \rightarrow \tau_y(C))$, where y is a variable different from x.

The intuition of the translation to FOL is that the formula $\tau_x(C)$ describes all domain elements d from $\Delta^{\mathcal{I}}$ that make the formula τ_x true if x is replaced by d. This clearly coincides with the interpretation of the concept description $C^{\mathcal{I}}$. The translation does not yield arbitrary FOL formulae, but formulae from the two-variable fragment [41] and the guarded fragment [40]. Both of which are known to be decidable.

Description Logics are closely related to modal logics (see e.g. [37, 21]). For instance, the DL $\mathcal{ALC}$ is a syntactic variant of the multimodal logic K, see [89]. The multimodal logic K introduces several box and diamond operators that are indexed with the name of the corresponding transition relation, which can be directly translated into $\mathcal{ALC}$ using role names corresponding to the transition relations.

Any $\mathcal{ALC}$ interpretation $\mathcal{I}$ can be viewed as a Kripke structure $K_{\mathcal{I}}$. The elements of the domain $w \in \Delta^{\mathcal{I}}$ correspond to possible worlds in $K_{\mathcal{I}}$. A propositional variable A is true in world w, iff $w \in A^{\mathcal{I}}$. There is a transition relation r in the Kripke structure from world w_1 to world w_2 iff $(w_1, w_2) \in r^{\mathcal{I}}$. Many theoretical results on reasoning in modal logics carry directly over to standard inferences in DLs due to this direct translation.

3 DL Reasoning

In this section we present reasoning methods for the DL reasoning problems defined in the last section: satisfiability and subsumption. These problems are decision problems and we devise decision procedures for them. Before we do so, we recall some general requirements that we would like to hold for such decision procedures. Such a procedure must be:

- *sound*, i.e., the positive answers should be correct;
- *complete*, i.e., the negative answers should be correct; and
- *terminating*, i.e., it should always give an answer in finite time.

Together these properties ensure that we always obtain an answer and that every given answer of the procedure is correct. These properties guarantee that applications built on top of these procedures are predictable and reliable. To employ the decision procedures in real world applications, we also would like our decision procedure to be

- *efficient*, i.e., it should be optimal w.r.t. the (worst-case) complexity of the problem, and
- *practical*, i.e., easy to implement and optimize, and behave well for application cases.

DL research has mostly been dedicated to design decision procedures that fulfill these requirements. The underlying techniques to realize reasoning procedures that we are considering in the following are the tableaux method for expressive DLs and completion for $\mathcal{EL}$.

3.1 Reasoning in Expressive DLs

By expressive DLs we refer to DLs that offer at least all Boolean constructors and that are thus closed under negation. For this kind of DLs, it is not necessary to design and implement different algorithms for the different reasoning problems introduced in the last section, since there exist polynomial time reductions, which only require the availability of the concept constructors conjunction and negation in the description language. For the TBox reasoning problems there are the following reductions:

- Subsumption can be be reduced in polynomial time to equivalence:

$$C \sqsubseteq_{\mathcal{T}} D \text{ iff } C \sqcap D \equiv_{\mathcal{T}} C.$$

- Equivalence can be be reduced in polynomial time to subsumption:

$$C \equiv_{\mathcal{T}} D \text{ iff } C \sqsubseteq_{\mathcal{T}} D \text{ and } D \sqsubseteq_{\mathcal{T}} C.$$

- Subsumption can be be reduced in polynomial time to (un)satisfiability:

$$C \sqsubseteq_{\mathcal{T}} D \text{ iff } C \sqcap \neg D \text{ is unsatisfiable w.r.t. } \mathcal{T}.$$

- Satisfiability can be be reduced in polynomial time to (non-)subsumption:

$$C \text{ is satisfiable w.r.t. } \mathcal{T} \text{ iff not } C \sqsubseteq_{\mathcal{T}} \bot.$$

For reasoning problems w.r.t. ABoxes (and TBoxes) there are similar polynomial time reductions:

- Satisfiability can be be reduced in polynomial time to consistency:

$$C \text{ is satisfiable w.r.t. } \mathcal{T} \text{ iff the ABox } \{C(a)\} \text{ is consistent w.r.t. } \mathcal{T}.$$

- The instance problem can be reduced in polynomial time to (in)consistency:

$$\mathcal{A} \models_{\mathcal{T}} C(a) \text{ iff } \mathcal{A} \cup \{\neg C(a)\} \text{ is inconsistent w.r.t. } \mathcal{T}.$$

- Consistency can be reduced in polynomial time to the (non-)instance problem:

$$\mathcal{A} \text{ is consistent w.r.t. } \mathcal{T} \text{ iff } \mathcal{A} \not\models_{\mathcal{T}} \bot(a).$$

With these reductions at hand, it suffices to investigate a reasoning procedure for one of the reasoning problems. In this section, we restrict ourselves to unfoldable TBoxes, i.e., TBoxes without GCIs and cyclic definitions. We present a tableau algorithm for deciding ABox consistency in this setting. Such a tableau-based algorithm tries to construct a model for the ABox by breaking down the concept descriptions in the knowledge base and inferring new constraints on the elements of this model. The algorithm either stops because all attempts to build a model failed due to obvious contradictions, or it stops with a "canonical" model.

In a first step of the consistency test, negation is treated by transforming the concept description from the knowledge base into *negation normal form (NNF)*. This normal form pushes all negations into the description until they occur only in front of concept names, using de Morgan' rules.

The $\rightarrow_\sqcap$-rule
Condition: $\mathcal{A}$ contains $(C_1 \sqcap C_2)(x)$, but not both $C_1(x)$ and $C_2(x)$.
Action: $\mathcal{A}' := \mathcal{A} \cup \{C_1(x), C_2(x)\}$.

The $\rightarrow_\sqcup$-rule
Condition: $\mathcal{A}$ contains $(C_1 \sqcup C_2)(x)$, but neither $C_1(x)$ nor $C_2(x)$.
Action: $\mathcal{A}' := \mathcal{A} \cup \{C_1(x)\}$, $\mathcal{A}'' := \mathcal{A} \cup \{C_2(x)\}$.

The $\rightarrow_\exists$-rule
Condition: $\mathcal{A}$ contains $(\exists r.C)(x)$, but there is no individual name z such that $C(z)$ and $r(x, z)$ are in $\mathcal{A}$.
Action: $\mathcal{A}' := \mathcal{A} \cup \{C(y), r(x, y)\}$ where y is an individual name not occurring in $\mathcal{A}$.

The $\rightarrow_\forall$-rule
Condition: $\mathcal{A}$ contains $(\forall r.C)(x)$ and $r(x, y)$, but it does not contain $C(y)$.
Action: $\mathcal{A}' := \mathcal{A} \cup \{C(y)\}$.

Fig. 1. Tableau rules of the consistency algorithm for $\mathcal{ALC}$.

Definition 12 ($\mathcal{ALC}$-negation normal form). *An $\mathcal{ALC}$-concept description is in $\mathcal{ALC}$-*negation normal form *(NNF) if the following rules have been applied exhaustively:*

$$\begin{array}{lll} \neg\bot \rightarrow \top & \neg(C \sqcap D) \rightarrow (\neg C \sqcup \neg D) & \neg(\exists r.C) \rightarrow (\forall r.\neg C) \\ \neg\top \rightarrow \bot & \neg(C \sqcup D) \rightarrow (\neg C \sqcap \neg D) & \neg(\forall r.C) \rightarrow (\exists r.\neg C) \\ & \neg\neg C \rightarrow C & \end{array}$$

A TBox or an ABox is in NNF, if all concept descriptions appearing in it are in NNF.

The *size* of an $\mathcal{ALC}$-concept description is the number of occurrences of all concept and role names that appear in the concept description. The size of a TBox is the sum of the sizes of all the concept descriptions appearing in the TBox. Similarly, the size of an ABox is the sum of all the concept descriptions appearing the concept assertions plus the number of role assertions. Transforming an $\mathcal{ALC}$-concept description into NNF yields an equivalent concept description, TBox or ABox of the same size.

Let $\mathcal{A}_0$ be an $\mathcal{ALC}$-ABox that is to be tested for consistency. In a first preprocessing step the definitions from the TBox are expanded.[2] More precisely, names of defined concepts are replaced by the right-hand sides of their definitions in the TBox. This replacement is done exhaustively until only names of primitive concepts appear in the ABox $\mathcal{A}_0$. Next, this ABox is transformed into NNF. In order to test consistency of the normalized $\mathcal{A}_0$, the algorithm applies *tableau rules* to this ABox until no more rules apply. The tableau rules for $\mathcal{ALC}$ are depicted in Fig. 1. Tableau rules in general are consistency preserving transformation rules.

The tableau rule $\rightarrow_\sqcup$ that handles disjunction is *nondeterministic*. It transforms a given ABox into two new ABoxes such that the original ABox is consistent if *one* of the new ABoxes is so. For this reason, we will consider finite sets of ABoxes $\mathcal{S} = \{\mathcal{A}_1, \ldots, \mathcal{A}_k\}$ instead of single ones. Such a set of ABoxes is *consistent* iff there is

[2] Recall, that we are dealing with unfoldable TBoxes (Def. 6).

some i, $1 \leq i \leq k$, such that $\mathcal{A}_i$ is consistent. A tableau rule of Fig. 1 is applied to a given finite set of ABoxes $\mathcal{S}$ as follows: it takes an element $\mathcal{A}$ of $\mathcal{S}$, and replaces it by one ABox $\mathcal{A}'$ or, in case of $\rightarrow_{\sqcup}$ by two ABoxes $\mathcal{A}'$ and $\mathcal{A}''$.

Definition 13 (Clash, complete ABox, closed ABox). *An ABox $\mathcal{A}$ contains a* clash *iff $\{A(x), \neg A(x)\} \subseteq \mathcal{A}$ for some individual name x and some concept name A. An ABox $\mathcal{A}$ is called*

- complete *iff none of the tableau rules of Fig. 1 applies to it, and*
- closed *if it contains a clash, and* open *otherwise.*

The *consistency algorithm for $\mathcal{ALC}$* proceeds in the following steps. It starts with the singleton set of ABoxes $\{\mathcal{A}_0\}$, and applies the rules from Fig. 1 in arbitrary order until no more rules apply. The algorithm returns "consistent" if the set $\widehat{\mathcal{S}}$ of ABoxes obtained by exhaustively applying the tableau rules contains an open ABox, and "inconsistent" otherwise.

For this procedure, one can show that it is sound, complete and terminating by examining the individual tableau rules. For termination, it is easy to see that each rule application is monotonic in the sense that every rule application extends the number of concept assertions for the individuals in $\mathcal{A}$ and it never removes elements from $\mathcal{A}$. Furthermore, each concept description that appears in $\mathcal{A}$ due to the application of the tableau rules is a sub-concept description of a concept description that appears already in the initial ABox $\mathcal{A}_0$. These two facts together imply that the application of tableau rules terminates. Completeness of the procedure can easily be seen from the definition of a clash. Soundness can be shown by showing local correctness of the individual tableau rules. Local correctness means that the rules preserve consistency, i.e., if $\widehat{\mathcal{S}}'$ is obtained from the finite set of ABoxes $\widehat{\mathcal{S}}$ by application of a transformation rule, then $\widehat{\mathcal{S}}$ is consistent iff $\widehat{\mathcal{S}}'$ is consistent.

Due to space limitations, we refer the reader to [2, 6] for the proofs for soundness, completeness and termination of the tableau algorithm for $\mathcal{ALC}$.

For general TBoxes, the tableau algorithm needs to be extended by a rule for treating GCIs and a more complex mechanism to ensure termination. For a given general TBox $\mathcal{T} = \{C_1 \sqsubseteq D_1, \ldots, C_n \sqsubseteq D_n\}$, it is easy to see that the general TBox consisting of the single GCI of the form

$$\top \sqsubseteq (\neg C_1 \sqcup D_1) \sqcap \ldots \sqcap (\neg C_n \sqcup D_n)$$

is equivalent to $\mathcal{T}$, i.e., they have the same models. Thus, reasoning for general TBoxes can be done by taking a general TBox that consists of a single GCI of the form $\top \sqsubseteq C$, where C is a concept description constructed from the GCIs as above. This GCI states that every element in the model belongs to C. To capture this in the tableau method, we add a new rule: the $\rightarrow_{\mathcal{T} \sqsubseteq C}$-rule adds the concept assertion $C(x)$ in case the individual name x occurs in the ABox $\mathcal{A}$, and $C(x)$ is not yet present in $\mathcal{A}$. Local correctness, soundness, and completeness of this procedure can easily be shown. However, the procedure does not terminate, due to cyclic axioms. To regain termination, cyclic computations need to be detected and the application of the $\rightarrow_{\exists}$-rule must be blocked. For two individuals a and b, we say that a is *younger* than b, if a was introduced by an

application of the $\rightarrow_\exists$-rule after b was already present in the ABox. The application of the $\rightarrow_\exists$-rule to an individual x is *blocked* by an individual y in an ABox $\mathcal{A}$ iff

- x is younger than y, and
- $\{C \mid C(x) \in \mathcal{A}\} \subseteq \{C \mid C(y) \in \mathcal{A}\}$.

The main idea underlying blocking is that the blocked individual x can use the role successors of y instead of generating new role successors.

The complexity of the consistency problem in $\mathcal{ALC}$ w.r.t. unfoldable TBoxes is PSpace-complete [92, 66]. In case general TBoxes are used, the complexity of testing consistency is ExpTime-complete [89]. For the DLs underlying the OWL standard the complexity of testing consistency is even higher. Reasoning in the DL underlying the OWL 1.0 standard $\mathcal{SHOIQ}$ is NExpTime-complete [98] and for the DL $\mathcal{SROIQ}$, which is the basis for the OWL 2 standard, it is even N2ExpTime [64].

3.2 Reasoning in $\mathcal{EL}$

Since the DL $\mathcal{EL}$ does neither offer negation nor the bottom concept, contradictions cannot be expressed and thus testing satisfiability is trivial in $\mathcal{EL}$. For testing subsumption in $\mathcal{EL}$, it was shown in [25] that reasoning can be done in polynomial time. This result was rather surprising. For the very similar DL $\mathcal{FL}_0$, which allows for value restrictions instead of existential restrictions, reasoning w.r.t. general TBoxes is ExpTime-complete [46]. For a collection of extensions of $\mathcal{EL}$ it was investigated, whether they have the same nice computational properties [26, 4, 5]. These investigations identified extensions of $\mathcal{EL}$ that allow for efficient classification. The DL $\mathcal{EL}^{++}$ extends $\mathcal{EL}$ with the bottom concept ($\bot$), nominals, a restricted form of concrete domains, and a restricted form of so-called role-value maps. For this DL, it was shown in [5] that almost all additions of other typical DL constructors to $\mathcal{EL}$ make subsumption w.r.t. general TBoxes ExpTime-complete. The DL $\mathcal{EL}^{++}$ is the closest DL to the OWL 2 EL profile.

Despite its limited expressivity, $\mathcal{EL}$ is highly relevant for practical applications. In fact, both the large medical ontology SNOMED CT[3] and the Gene Ontology[4] can be expressed in $\mathcal{EL}$.

3.3 Subsumption in $\mathcal{EL}$

The polynomial time algorithm for computing subsumption w.r.t. a general TBox actually performs classification of the whole TBox, i.e., it computes the subsumption relationships between all named concepts of a given TBox simultaneously. This algorithm proceeds in four steps:

1. Normalize the TBox.
2. Translate the normalized TBox into completion sets.
3. Complete these sets using completion rules.
4. Read off the subsumption relationships from the normalized graph.

[3] http://www.ihtsdo.org/snomed-ct/
[4] http://www.geneontology.org/

NF1	$C \sqcap \hat{D} \sqsubseteq E \longrightarrow \{\ \hat{D} \sqsubseteq A, C \sqcap A \sqsubseteq E\ \}$
NF2	$\exists r.\hat{C} \sqsubseteq D \longrightarrow \{\ \hat{C} \sqsubseteq A, \exists r.A \sqsubseteq D\ \}$
NF3	$\hat{C} \sqsubseteq \hat{D} \longrightarrow \{\ \hat{C} \sqsubseteq A, A \sqsubseteq \hat{D}\ \}$
NF4	$B \sqsubseteq \exists r.\hat{C} \longrightarrow \{\ B \sqsubseteq \exists r.A, A \sqsubseteq \hat{C}\ \}$
NF5	$B \sqsubseteq C \sqcap D \longrightarrow \{\ B \sqsubseteq C, B \sqsubseteq D\ \}$

where $\hat{C}, \hat{D}$ are complex concept descriptions and A is a new concept name.

Fig. 2. $\mathcal{EL}$ normalization rules

The normal form for $\mathcal{EL}$-TBoxes required in the first step is defined as follows.

Definition 14 (Normal form for $\mathcal{EL}$-TBoxes). *An $\mathcal{EL}$-TBox $\mathcal{T}$ is in* normal form *if all concept inclusions have one of the following forms:*

$$A_1 \sqsubseteq B, \quad A_1 \sqcap A_2 \sqsubseteq B, \quad A_1 \sqsubseteq \exists r.A_2 \quad \text{or} \quad \exists r.A_1 \sqsubseteq B,$$

where A_1, A_2 and B are concept names appearing in $\mathcal{T}$ or the top-concept $\top$.

Any $\mathcal{EL}$-TBox $\mathcal{T}$ can be transformed into a normalized TBox $\mathcal{T}'$ by simply introducing new concept names. $\mathcal{EL}$-TBoxes can be transformed into normal form by applying the normalization rules displayed in Fig. 2 exhaustively. These rules replace the GCI on the left-hand side of the rule with the set of GCIs on the right-hand side of the rule. The idea behind the normalization rules is to introduce names for complex sub-concept descriptions. It suffices to obtain a TBox that is a subsumption-equivalent TBox to the original one, i.e., the original and the normalized TBox capture the same subsumption relationships for the named concepts from the original TBox. Thus it suffices to introduce the new concept names with GCIs instead of equivalences. The transformation into normal form can be done in linear time.

The completion algorithm works on a data-structure called *completion sets*. There are two kinds of completion sets used in the algorithm:

- $S(A)$ for each concept name A mentioned in the normalized TBox, and
- $S(A, r)$ for each concept name A and role name r mentioned in the normalized TBox.

Both kinds of completion sets contain concept names and $\top$. By $\mathsf{S}_\mathcal{T}$ we denote the set containing all completion sets of the TBox $\mathcal{T}$. In the completion algorithm, the completion sets are initialized as follows:

- $S(A) := \{A, \top\}$ for each concept name A mentioned in the normalized TBox, and
- $S(A, r) := \emptyset$ for each concept name A and role name r mentioned in the normalized TBox.

CR1	If $C' \sqsubseteq D \in \mathcal{T}$, $C' \in S(C)$, and $D \notin S(C)$ then add D to $S(C)$.
CR2	If $C_1 \sqcap C_2 \sqsubseteq D \in \mathcal{T}$, $C_1, C_2 \in S(C)$, and $D \notin S(C)$ then add D to $S(C)$.
CR3	If $C' \sqsubseteq \exists r.D \in \mathcal{T}$, $C' \in S(C)$, and $D \notin S(C, r)$ then add D to $S(C, r)$.
CR4	If $\exists r.D' \sqsubseteq E \in \mathcal{T}$, $D \in S(C, r)$, $D' \in S(D)$, and $E \notin S(C)$ then add E to $S(C)$.

Fig. 3. $\mathcal{EL}$ completion rules

The intuition is that the completion rules make implicit subsumption relationships explicit in the following sense:

- $B \in S(A)$ implies that $A \sqsubseteq_{\mathcal{T}} B$, i.e., $S(A)$ contains only subsumers of A, and
- $B \in S(A, r)$ implies that $A \sqsubseteq_{\mathcal{T}} \exists r.B$, i.e., $S(A, r)$ contains only concept names B s.t. A is subsumed by $\exists r.B$.

In fact, it can be shown that these properties of the completion sets are *invariants* and thus do not change during completion. Clearly, this holds for the initial elements of the completion. After initialization all completion sets in $\mathsf{S}_{\mathcal{T}}$ are extended by applying the completion rules that are shown in Fig. 3 exhaustively, i.e., until no more rule applies. It is easy to see that the rules preserve the above invariants. In each of the rules the last condition ensures that the rule is only applied once to the same concepts and completion sets. The first rule **CR1** propagates the transitivity of subsumption. The second **CR2** ensures that if a conjunction implies a concept C w.r.t. $\mathcal{T}$ and the conjuncts are already in the completion set of a concept, then C has to be in that completion set as well. The rule **CR3** is applicable if a concept name implies an existential restriction w.r.t. $\mathcal{T}$ and this concept name is contained in the completion set $S(C)$, then the existential restriction is implied by C as well. The most complicated rule is **CR4**. The axiom $\exists r.D' \sqsubseteq E \in \mathcal{T}$ implies $\exists r.D' \sqsubseteq_{\mathcal{T}} E$, and the assumption that the invariants are satisfied before applying the rule yields $D \sqsubseteq_{\mathcal{T}} D'$ and $C \sqsubseteq_{\mathcal{T}} \exists r.D$. The subsumption relationship $D \sqsubseteq_{\mathcal{T}} D'$ then implies $\exists r.D \sqsubseteq_{\mathcal{T}} \exists r.D'$. By applying transitivity of the subsumption relation $\sqsubseteq_{\mathcal{T}}$, we obtain $C \sqsubseteq_{\mathcal{T}} E$.

Once the completion process has terminated, the subsumption relation between two named concepts A and B can be tested by checking whether $B \in S(A)$. The fact that subsumption in $\mathcal{EL}$ w.r.t. general TBoxes can be decided in polynomial time follows from the following statements:

1. Rule application terminates after a polynomial number of steps.
2. If no more rules are applicable, then $A \sqsubseteq_{\mathcal{T}} B$ iff $B \in S(A)$.

The first statement holds, since the number of completion sets, of the kind $S(A)$ is linear in size of the TBox. In addition, the number of completion set of the kind $S(A, r)$ is quadratic in the size of $\mathcal{T}$. The size of the completion sets is bounded by the number of concept names and role names, and each rule application extends at least one label.

Theorem 1. *Subsumption in $\mathcal{EL}$ is polynomial w.r.t. general TBoxes.*

This nice computational property transfers also to $\mathcal{EL}^{++}$ [5], the DL corresponding closest to the OWL 2 EL profile.

The first implementation of the subsumption algorithm for $\mathcal{EL}$ sketched above is the CEL system [11, 71]. This system showed that the classification of the very large knowledge bases can be done in runtime acceptable for practical applications. For instance, classifying the knowledge base SNOMED CT, which contains more than 300.000 axioms takes less than half an hour and classification of the Gene Ontology, which contains more than 20.000 axioms, takes only 6 seconds [12].

4 Explanation of Reasoning Results

DL knowledge bases often contain thousands of axioms and have a complex structure due to the use of GCIs. These knowledge bases are developed by users who are experts in the domain to be modeled, but have little expertise in knowledge representation or logic. For this sort of applications, it is necessary that the development process of the knowledge base is supported by automated services implemented in the DL system.

Classical DL reasoning systems can detect that a certain consequence holds, such as an inconsistency or a subsumption relation, but they give no evidence *why* it holds. The reasoning service explanation facilitates better understanding of the knowledge base and gives a starting point to resolve an unwanted consequence in the knowledge base. For instance, the SNOMED ontology contains the subsumption relation:

$$\mathsf{Amputation\text{-}of\text{-}Finger} \sqsubseteq \mathsf{Amputation\text{-}of\text{-}Arm}.$$

A user who wants to correct this, faces the task of finding the axioms responsible for this unintended subsumption relation among 350.000 others. Clearly, automated support is needed for this task. A first step towards providing such support was described in [90], where an algorithm for computing all minimal subsets of a given knowledge base that have a given consequence is described. This approach was extended to expressive DLs in [83].

For a TBox $\mathcal{T}$ and a consequence c an *explanation* points to the "source" of the consequence, which is a subset of $\mathcal{T}$ that contributes to the consequence c. We call a *minimal axiom set* (MinA) a minimal subset (w.r.t. size) of a TBox $\mathcal{T}$, that has a certain consequence. *Axiom pinpointing* is the process of computing MinAs.

Example 1. Consider the following TBox:

$$\begin{aligned}
\mathcal{T}_{ex} = \{ \quad & \mathsf{Cat} \sqsubseteq \exists\, \mathsf{has\text{-}parent}.\, \mathsf{Cat}, && \text{I} \\
& \mathsf{Cat} \sqsubseteq \mathsf{Pet}, && \text{II} \\
& \exists\, \mathsf{has\text{-}parent}.\mathsf{Pet} \sqsubseteq \mathsf{Animal}, && \text{III} \\
& \mathsf{Pet} \sqsubseteq \mathsf{Animal} \quad \} && \text{IV}
\end{aligned}$$

For the TBox $\mathcal{T}_{ex}$, we find the consequence $\mathsf{Cat} \sqsubseteq_{\mathcal{T}_{ex}} \mathsf{Animal}$. The consequence holds since axiom I says that cats are pets and pets are in turn animals by axiom IV. This

consequence also follows from $\mathcal{T}_{ex}$ by using axiom I and axiom II, which together say that a cat has a parent that is a pet. Now from this together with axiom III it, follows that cats are animals. Thus, the one consequence has several MinAs, namely: $\{\text{I, IV}\}$ and $\{\text{I, II, III}\}$.

It turns out that there may be exponentially many MinAs, which shows that an algorithm for computing *all* MinAs needs exponential time in the size of the input TBox. In order to obtain an explanation for a consequence, we need to compute one single MinA of the consequence. There are two general approaches for pinpointing, i.e., computing a MinA of a consequence:

Black box approach, which uses a DL reasoner as an oracle, i.e, it repetitively queries the reasoner to compute a MinA.

Glass box approach, which modifies the internals of a DL reasoner s.t. it yields a MinA directly when computing an inference.

While the black box approach is independent of the reasoner, the glass box approach needs to be tailored to the reasoning method in use. We examine the black box approach first, which is the method of choice for expressive DLs, then we discuss the glass box approach for completion-based reasoning in $\mathcal{EL}$.

The task of computing explanations has also been considered in other research areas. For example, in the SAT community, people have considered the problem of computing minimally unsatisfiable subsets of a set of propositional formulae. Approaches for computing these sets developed there include algorithms that call a SAT solver as a black box [65, 20] but also algorithms that extend a resolution-based SAT solver directly [34, 103].

4.1 Black Box Method for Pinpointing

Assume we want to perform pinpointing for the consequence $A \sqsubseteq B$ w.r.t. the TBox $\mathcal{T}$. The basic idea underlying the black box method is a kind of uninformed search: Given a TBox $\mathcal{T}$ and the consequence $A \sqsubseteq B$: simply remove the first axiom from the TBox $\mathcal{T}$ and test whether the consequence still holds. If so, continue with the second axiom. If the consequence does *not* follow from the TBox with the first axiom removed, put the axiom back to the TBox and then test the second axiom. This naive method always performs as many subsumption tests as the number of axioms in the TBox. Since MinAs are often quite small, this is not a feasible method for very large TBoxes.

A more efficient method would not proceed axiom-wise, but first compute a not necessarily minimal subset of the TBox from which the consequence follows and then minimize this set using the naive procedure. This approach is only feasible if the algorithm for the first step produces fairly small sets of axioms and is efficient.

The black box method is independent of the DL in use and can be used to compute explanations for any DL, provided there is a DL reasoner for the DL and the consequence in question. This method can easily be implemented on top of a DL reasoner and does not require to change the internal structure of the reasoner. This is the reason why most implementations of pinpointing are based on the black box approach.

For $\mathcal{EL}$ the black box pinpointing algorithm has been implemented in the DL reasoning system CEL [16, 19, 97]. For a variant of the medical knowledge base GALEN [87] with 4000 axioms the overall run-time for computing a MinA with the non-naive method took 9:45 min. In contrast the naive method took seven hours for the same task. The first implementation of the black box method for pinpointing was done for the ontology editor SWOOP [62] based on the methods described in [83]. A more recent implementation of black box pinpointing was done in the ontology editor PROTÉGÉ. This implementation allows pinpointing even for *parts* of axioms that contribute to deriving a consequence [47].

4.2 Glass Box Pinpointing for $\mathcal{EL}$

The glass box approach for computing an explanation depends on the DL used and the reasoning method employed. It requires that the internals of a reasoner are modified by adding label sets to the reasoning procedure that collect the relevant axioms already during the computation of the consequence. For $\mathcal{EL}$, we modify the completion algorithm for subsumption from Section 3.3 to compute one explanation for a subsumption relationship. To this end, we annotate every element in the completion sets in S with a monotone Boolean formula that captures the MinAs.[5] The glass box algorithm for $\mathcal{EL}$ was described in [15] and extended in [16].

The *basic labeling* assigns to every GCI $t \in \mathcal{T}$ a unique propositional variable $lab(t)$ as a label. By $lab(\mathcal{T})$ we denote the set of all propositional variables labeling GCIs in the TBox $\mathcal{T}$. Now, a *monotone Boolean formula over $lab(\mathcal{T})$* is a Boolean formula using

- (some of) the variables in $lab(\mathcal{T})$, and
- only the connectives $\wedge$, $\vee$ and $true$ for truth.

Its propositional *valuation* (denoted ν) is the set of propositional variables that make the formula true when they are assigned the value true. For a valuation $\nu \subseteq lab(\mathcal{T})$, let $\mathcal{T}_\nu := \{t \in \mathcal{T} \mid lab(\mathcal{T}) \in \nu\}$. The idea is that the valuation characterizes a combination of axiom labels. These labels are mapped back to the actual axioms from the TBox $\mathcal{T}$ by $\mathcal{T}_\nu$.

Definition 15 (Pinpointing formula). *Let $\mathcal{T}$ be an $\mathcal{EL}$-TBox and A and B concept names occurring in $\mathcal{T}$. The* monotone Boolean formula ϕ over $lab(\mathcal{T})$ *is a pinpointing formula for $\mathcal{T}$ w.r.t. $A \sqsubseteq_\mathcal{T} B$, if the following holds for every valuation $\nu \subseteq lab(\mathcal{T})$:*

$$A \sqsubseteq_{\mathcal{T}_\nu} B \text{ iff } \nu \text{ satisfies } \phi.$$

Consider Example 1 again. Take $lab(\mathcal{T}_{ex}) := \{\mathrm{I}, \mathrm{II}, \mathrm{III}, \mathrm{IV}\}$ as the set of propositional variables, then $\mathrm{II} \wedge (\mathrm{IV} \vee (\mathrm{I} \wedge \mathrm{III}))$ is a pinpointing formula for $\mathcal{T}_{ex}$ w.r.t. $A \sqsubseteq_{\mathcal{T}_{ex}} B$.

Lemma 1. *Let ϕ be a pinpointing formula for the TBox $\mathcal{T}$ w.r.t. $A \sqsubseteq_\mathcal{T} B$. If valuations are ordered by set inclusions, then*

[5] This method for generating explanations was first applied for default reasoning in [8].

$$M = \{\mathcal{T}_\nu \mid \nu \textit{ is a minimal valuation satisfying } \phi\}$$

is the set of all *MinAs for* $\mathcal{T}$ *w.r.t.* $A \sqsubseteq_\mathcal{T} B$.

Proof. We need to show the following claims:

1. M contains only MinAs.
2. There is no MinA m_1 s.t. $m_1 \notin M$.

Show claim 1.:
For each set of axioms $m \in M$ there is a valuation ν_m s.t. $\nu_m = lab(m)$, which is minimal in size and that satisfies ϕ. Since ϕ is satisfied, $A \sqsubseteq_\mathcal{T} B$ holds. Since ν_m is minimal there is no subset of ν_m satisfying ϕ, and thus m is a MinA.
Show claim 2.:
Assume m_1 is a MinA for $\mathcal{T}$ w.r.t. $A \sqsubseteq_\mathcal{T} B$ and $m_1 \notin M$. Since m_1 is a MinA, m_1 is minimal and $A \sqsubseteq_{m_1} B$ holds. Let ν_{m_1} be the valuation $\nu_{m_1} = lab(m_1)$. From $A \sqsubseteq_\mathcal{T} B$ follows ν_{m_1} satisfies the pinpointing formula ϕ. Thus, m_1 induces a minimal valuation satisfying ϕ, which is a contradiction to $m_1 \notin M$. ❑

Lemma 1 guarantees that it is enough to compute the pinpointing formula to obtain *all* MinAs, i.e., explanations for the consequence in question. However, to obtain one MinA from the pinpointing formula, one can transform the pinpointing formula into disjunctive normal form, remove those disjuncts that are implied by other disjuncts and then pick one disjunct as the explanation.

Next, we describe the computation algorithm for pinpointing formulae in $\mathcal{EL}$ based on completion. Again, we want to explain $A \sqsubseteq B$ w.r.t. the $\mathcal{EL}$-TBox $\mathcal{T}$. Since the completion algorithm starts by normalizing the TBox, we need to introduce the labels for the original TBox and labels for the normalized TBox $\mathcal{T}'$ as well. The labels of the normalized TBox $\mathcal{T}'$ need to "keep track" of the corresponding axioms in the original TBox.

The completion procedure needs to be adapted to propagate the labels and to construct the pinpointing formula. To this end, each element of the completion sets, say $X \in S(A)$, is labelled with a monotone Boolean formula: $lab(A, X)$. The initial elements of the completions sets $A \in S(A)$ and $\top \in S(A)$ are labelled with $true$, i.e., $lab(A, A) = lab(A, \top) = true$ for all concept names appearing in $\mathcal{T}$. Now, we need to modify the completion rules from Fig. 3. Let the precondition of a completion rule **CRi** be satisfied for a set of completion sets $\mathsf{S}_{\mathcal{T}'}$ w.r.t. the TBox $\mathcal{T}'$. The modified rule collects the labels of those GCIs and completion sets that make the rule **CRi** applicable. Let ϕ be the conjunction of :

- labels of GCIs in $\mathcal{T}'$ that appear in the precondition of **CRi**, and
- labels of elements in completion sets in $\mathsf{S}_{\mathcal{T}'}$ that appear in the precondition of **CRi**.

The conjunction collected in ϕ needs to be propagated to the consequence of the rule **CRi**. If the completion set element in the consequence of **CRi** is *not* in $\mathsf{S}_{\mathcal{T}'}$, then it is added with label ϕ. In case the consequence of **CRi** *is* already in $\mathsf{S}_{\mathcal{T}'}$ and has the label ψ, the completion algorithm has derived the consequence again. In this case, ψ and ϕ are compared. If $\psi \wedge \phi \not\equiv \psi$, the consequence of **CRi** is derived in an alternative way and

the label of this consequence is changed to $\phi \vee \psi$. The new label of the consequence is a more general Boolean formula. If $\psi \wedge \phi \equiv \psi$, then ϕ implies ψ. In this case the rule **CRi** is not applied.

Example 2. Consider Example 1 again. To compute the pinpointing formula for $\mathsf{Cat} \sqsubseteq_{\mathcal{T}_{ex}} \mathsf{Animal}$, the set of completion sets $\mathsf{S}_{\mathcal{T}_{ex}}$ is initialized as follows:

$$\begin{aligned}\mathsf{S}_{\mathcal{T}_{ex}} = \{\ & (\mathsf{Cat}, \top)^{true}, (\mathsf{Cat}, \mathsf{Cat})^{true}, \\ & (\mathsf{Pet}, \top)^{true}, (\mathsf{Pet}, \mathsf{Pet})^{true}, \\ & (\mathsf{Animal}, \top)^{true}, (\mathsf{Animal}, \mathsf{Animal})^{true} \ \}.\end{aligned}$$

Then we can apply the modified rules:

- Using axiom II: $\mathsf{Cat} \sqsubseteq \mathsf{Pet} \in \mathcal{T}_{ex}$ and $(\mathsf{Cat}, \mathsf{Cat})^{true} \in \mathsf{S}_{\mathcal{T}_{ex}}$, add $(\mathsf{Cat},\mathsf{Pet})^{\mathrm{II} \wedge true}$ to $\mathsf{S}_{\mathcal{T}_{ex}}$.
- Using axiom I: $\mathsf{Cat} \sqsubseteq \exists\, \mathsf{has\text{-}parent}.\, \mathsf{Cat} \in \mathcal{T}_{ex}$ and $(\mathsf{Cat}, \mathsf{Cat})^{true} \in \mathsf{S}_{\mathcal{T}_{ex}}$, add $(\mathsf{Cat}, \mathsf{has\text{-}parent}, \mathsf{Pet})^{\mathrm{I} \wedge true}$ to $\mathsf{S}_{\mathcal{T}_{ex}}$.
- Using axiom IV: $\mathsf{Pet} \sqsubseteq \mathsf{Animal} \in \mathcal{T}_{ex}$ and $(\mathsf{Cat},\mathsf{Pet})^{\mathrm{II} \wedge true} \in \mathsf{S}_{\mathcal{T}_{ex}}$, add $(\mathsf{Cat}, \mathsf{Animal})^{\mathrm{II} \wedge \mathrm{IV} \wedge true}$ to $\mathsf{S}_{\mathcal{T}_{ex}}$.
- Using axiom III: $\exists\, \mathsf{has\text{-}parent}.\mathsf{Pet} \sqsubseteq \mathsf{Animal} \in \mathcal{T}_{ex}$ and $\{(\mathsf{Cat}, \mathsf{Pet})^{\mathrm{II} \wedge true}, (\mathsf{Cat}, \mathsf{has\text{-}parent}, \mathsf{Pet})^{\mathrm{I} \wedge true})\} \subset \mathsf{S}_{\mathcal{T}_{ex}}$, modify $(\mathsf{Cat}, \mathsf{Animal})^{\mathrm{II} \wedge \mathrm{IV} \wedge true}$ to $(\mathsf{Cat}, \mathsf{Animal})^{(\mathrm{II} \wedge \mathrm{IV} \wedge true) \vee (\mathrm{III} \wedge \mathrm{II} \wedge \mathrm{I} \wedge true)}$.

Now, $lab(\mathsf{Cat}, \mathsf{Animal}) = (\mathrm{II} \wedge \mathrm{IV}) \vee (\mathrm{III} \wedge \mathrm{II} \wedge \mathrm{I})$ is the pinpointing formula for $\mathcal{T}_{ex}$ w.r.t. $\mathsf{Cat} \sqsubseteq_{\mathcal{T}_{ex}} \mathsf{Animal}$.

The modified completion algorithm always terminates, but not necessarily in polynomial time due to the possibility of repeated generalization of the label. Testing equivalence of monotone Boolean formulae is an NP-complete problem. However, given formulae over n propositional variables whose size is exponential in n, equivalence can be tested in time exponential in n. Thus, there are at most exponentially many rule applications and each of them takes at most exponential time. This yields an exponential time bound for the execution of the pinpointing algorithm.

However, the set of completion sets S obtained by the described process is identical to the one obtained by the unmodified algorithm. After the modified completion algorithm has terminated, the label $lab(A, B)$ is a pinpointing formula for $\mathcal{T}$ w.r.t. $A \sqsubseteq_{\mathcal{T}} B$.

Theorem 2. *Given an $\mathcal{EL}$-TBox $\mathcal{T}$ in normal form, the pinpointing algorithm terminates in time exponential in the size of $\mathcal{T}$. After termination, the resulting set of completion sets $\mathsf{S}_{\mathcal{T}}$ satisfies the following two properties for all concept names A,B occurring in $\mathcal{T}$:*

1. *$A \sqsubseteq_{\mathcal{T}} B$ iff $(S(A), B) \in \mathsf{S}_{\mathcal{T}}$, and*
2. *lab(A,B) is a pinpointing formula for $\mathcal{T}$ w.r.t $A \sqsubseteq_{\mathcal{T}} B$.*

This result was shown in [16] for the DL $\mathcal{EL}^{++}$. In the example, the TBox $\mathcal{T}_{ex}$ is already in normal form. In the general case, the TBox needs to be normalized and the pinpointing formula obtained by the modified completion needs to reconstruct the labels for the original axioms from the label of the normalized axioms.

The propositional variables from the normalized TBox in ϕ are replaced with those of the original one. More precisely, each label of a normalized GCI is replaced by the disjunction of its source GCIs. Once the de-normalized pinpointing formula is obtained, it is transformed into disjunctive normal form. One disjunct of this formula yields a MinA and thus an explanation of the consequence. To sum up, the *pinpointing extension* of the $\mathcal{EL}$ subsumption algorithm proceeds in the following steps:

1. Label all axioms in $\mathcal{T}$.
2. Normalize $\mathcal{T}$ according the rules from Fig. 2.
3. Label each axiom in the normalized TBox $\mathcal{T}'$ and keep the source GCI of every normalized GCI.
4. Apply the completion rules from Fig. 3 *modified* as described.
5. De-normalize the pinpointing formula.
6. Build the disjunctive normal form.
7. Pick one disjunct as explanation.

Note that the transformation into disjunctive normal form may cause an exponential blow-up, which means that, in some cases, the pinpointing formula provides us with a compact representation of the set of all MinAs. Also note that this blow-up is not in the size of the pinpointing formula but rather in the number of variables. Thus, if the size of the pinpointing formula is already exponential in the size of the TBox $\mathcal{T}$, computing all MinAs from it is still "only" exponential in the size of $\mathcal{T}$.

The glass box approach for pinpointing has also been investigated for more expressive DLs such as $\mathcal{ALC}$ in [72]. A more general view on tableaux and pinpointing was taken in [14].

We presented methods to obtain an explanation for a consequence. In order to actually repair a DL knowledge base, it is necessary to alleviate *all* causes of an unwanted consequence. In order to support users to repair a knowledge base, all MinAs need to be computed. The glass box method for $\mathcal{EL}$ computes all MinAs and can be employed for knowledge base repair directly. For the black box approach, a method for obtaining all MinAs is described in [90, 60]. This method computes the first MinA by the algorithms described above and then employs a method based on *hitting sets* to obtain the remaining MinAs.

The mechanism of pinpointing is not only useful for explanation or repair of DL knowledge bases. Access restrictions to knowledge bases can be supported as well [9]. If a user only has access to a part of the ontology, it is not obvious whether certain consequences can be accessed by the user as well. By computing all MinAs for the consequence, it can be tested whether the consequence follows from the accessible part alone. In that case access to the consequence does not violate the access restrictions.

Acknowledgement This article is based on the Description Logic tutorial by the author, which she taught at the 2009 Masters Ontology Spring School organized by the Meraka Institute in Tshwane (Pretoria), South Africa and it is based on the course material by Franz Baader at the 2009 Reasoning Web Summer School, see [2]. The author would like to thank the anonymous reviewers and Marcel Lippmann for valuable comments on earlier versions of this paper.

References

1. F. Baader. Description logic terminology. In *[6]*, pages 485–495. Cambridge University Press, 2003.
2. F. Baader. Description logics. In *Proceedings of Reasoning Web: Semantic Technologies for Information Systems*, volume 5689 of *Lecture Notes in Computer Science*, pages 1–39, 2009.
3. F. Baader, A. Bauer, P. Baumgartner, A. Cregan, A. Gabaldon, K. Ji, K. Lee, D. Rajaratnam, and R. Schwitter. A novel architecture for situation awareness systems. In M. Giese and A. Waaler, editors, *Proc. of the 18th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (Tableaux 2009)*, volume 5607 of *Lecture Notes in Computer Science*, pages 77–92. Springer-Verlag, 2009.
4. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI-05)*, Edinburgh, UK, 2005. Morgan-Kaufmann Publishers.
5. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope further. In K. Clark and P. F. Patel-Schneider, editors, *In Proc. of the OWLED Workshop*, 2008.
6. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
7. F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H.-J. 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.
8. F. Baader and B. Hollunder. Embedding defaults into terminological knowledge representation formalisms. In *Proc. of the 3rd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-92)*, pages 306–317. Morgan Kaufmann, Los Altos, 1992.
9. F. Baader, M. Knechtel, and R. Peñaloza. A generic approach for large-scale ontological reasoning in the presence of access restrictions to the ontology's axioms. *Proc. of the 8th International Semantic Web Conference (ISWC 2009)*, volume 5823 of *Lecture Notes in Computer Science*, pages 49–64, 2009.
10. F. Baader, R. Küsters, and R. Molitor. Computing least common subsumer in description logics with existential restrictions. In T. Dean, editor, *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI-99)*, pages 96–101, 1999. Morgan Kaufmann, Los Altos.
11. F. Baader, C. Lutz, and B. Suntisrivaraporn. CEL—a polynomial-time reasoner for life science ontologies. In U. Furbach and N. Shankar, editors, *Proc. of the 3rd Int. Joint Conf. on Automated Reasoning (IJCAR-06)*, volume 4130 of *Lecture Notes In Artificial Intelligence*, pages 287–291. Springer-Verlag, 2006. CEL download page: `http://lat.inf.tu-dresden.de/systems/cel/`.
12. F. Baader, C. Lutz, and B. Suntisrivaraporn. Is tractable reasoning in extensions of the description logic $\mathcal{EL}$ useful in practice? In *Journal of Logic, Language and Information, Special Issue on Method for Modality (M4M)*, 2007.
13. F. Baader, C. Lutz, and A.-Y. Turhan. Small is again beautiful in description logics. *KI – Künstliche Intelligenz*, 24(1):25–33, April 2010.
14. F. Baader and R. Peñaloza. Axiom pinpointing in general tableaux. *Journal of Logic and Computation*, 20(1):5–34, 2010. Special Issue: Tableaux and Analytic Proof Methods.
15. F. Baader, R. Peñaloza, and B. Suntisrivaraporn. Pinpointing in the description logic $\mathcal{EL}$. In D. Calvanese, E. Franconi, V. Haarslev, D. Lembo, B. Motik, S. Tessaris, and A.-Y. Turhan, editors, *Proc. of the 2007 Description Logic Workshop (DL 2007)*, CEUR-WS, 2007.

16. F. Baader, R. Peñaloza, and B. Suntisrivaraporn. Pinpointing in the description logic $\mathcal{EL}^+$. In *Proc. of the 30th German Annual Conf. on Artificial Intelligence (KI'07)*, volume 4667 of *Lecture Notes In Artificial Intelligence*, pages 52–67, Osnabrück, Germany, 2007. Springer.
17. F. Baader and U. Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69:5–40, 2001.
18. F. Baader, B. Sertkaya, and A.-Y. Turhan. Computing the least common subsumer w.r.t. a background terminology. *Journal of Applied Logics*, 2007.
19. F. Baader and B. Suntisrivaraporn. Debugging SNOMED CT using axiom pinpointing in the description logic $\mathcal{EL}^+$. In *Proc. of the International Conference on Representing and Sharing Knowledge Using SNOMED (KR-MED'08)*, Phoenix, Arizona, 2008.
20. J. Bailey and P. J. Stuckey. Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization. In M. V. Hermenegildo and D. Cabeza, editors, *InProc. of Practical Aspects of Declarative Languages, 7th International Symposium (PADL 2005), USA*, LNCS, pages 174–186, 2005.
21. P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, MA, USA, 2001.
22. A. Borgida and P. F. Patel-Schneider. A semantics and complete algorithm for subsumption in the CLASSIC description logic. *Journal of Artificial Intelligence Research*, 1:277–308, 1994.
23. R. J. Brachman, A. Borgida, D. L. McGuinness, and L. Alperin Resnick. The CLASSIC knowledge representation system, or, KL-ONE: the next generation. Preprints of the Workshop on Formal Aspects of Semantic Networks, Two Harbors, Cal., 1989.
24. R. J. Brachman and H. J. Levesque. *Readings in Knowledge Representation*. Morgan Kaufmann, Los Altos, 1985.
25. S. Brandt. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In R. L. de Mantáras and L. Saitta, editors, *Proc. of the 16th European Conf. on Artificial Intelligence (ECAI-04)*, pages 298–302. IOS Press, 2004.
26. S. Brandt. Reasoning in $\mathcal{ELH}$ w.r.t. general concept inclusion axioms. LTCS-Report LTCS-04-03, Chair for Automata Theory, Institute for Theoretical Computer Science, Dresden University of Technology, Germany, 2004. See http://lat.inf.tu-dresden.de/research/reports.html.
27. S. Brandt, R. Küsters, and A.-Y. Turhan. Approximation and difference in description logics. In D. Fensel, D. McGuinness, and M.-A. Williams, editors, *Proc. of the 8th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-02)*, San Francisco, CA, 2002. Morgan Kaufmann Publishers.
28. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. DL-Lite: Tractable description logics for ontologies. In M. M. Veloso and S. Kambhampati, editors, *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI'05)*, pages 602–607. AAAI Press/The MIT Press, 2005.
29. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 260–270, 2006.
30. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *Journal of Automated Reasoning*, 39(3):385–429, 2007.
31. D. Calvanese and G. D. Giacomo. Expressive description logics. In *[6]*, pages 178–218. Cambridge University Press, 2003.
32. B. Cuenca Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Modular reuse of ontologies: Theory and practice. *Journal of Artificial Intelligence Research*, 31:273–318, 2008.

33. B. Cuenca Grau, Y. Kazakov, I. Horrocks, and U. Sattler. A logical framework for modular integration of ontologies. In *Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI-07)*, pages 298–303, 2007.
34. G. Davydov, I. Davydova, and H. K. Büning. An efficient algorithm for the minimal unsatisfiability problem for a subclass of cnf. *Ann. Math. Artif. Intell.*, 23(3-4):229–245, 1998.
35. F. M. Donini, S. Colucci, T. Di Noia, and E. Di Sciascio. A tableaux-based method for computing least common subsumers for expressive description logics. In *Proc. of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09)*, pages 739–745. AAAI, July 2009.
36. T. Eiter, C. Lutz, M. Ortiz, and M. Simkus. Query answering in description logics with transitive roles. In *Proc. of the 21st International Joint Conference on Artificial Intelligence IJCAI09*. AAAI Press, 2009.
37. M. Fitting. Basic modal logic. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 1, pages 365–448. Oxford Science Publications, 1993.
38. S. Ghilardi, C. Lutz, and F. Wolter. Did I damage my ontology? a case for conservative extensions in description logics. In P. Doherty, J. Mylopoulos, and C. Welty, editors, *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-06)*, pages 187–197. AAAI Press, 2006.
39. B. Glimm, I. Horrocks, C. Lutz, and U. Sattler. Conjunctive query answering for the description logic $\mathcal{SHIQ}$. In M. M. Veloso, editor, *Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI-07)*, pages 399–404, 2007.
40. E. Grädel. On the restraining power of guards. *Journal of Symbolic Logic*, 64:1719–1742, 1999.
41. E. Grädel, P. G. Kolaitis, and M. Y. Vardi. On the decision problem for two-variable first-order logic. *Bulletin of Symbolic Logic*, 3(1):53–69, 1997.
42. V. Haarslev and R. Möller. High performance reasoning with very large knowledge bases: A practical case study. In B. Nebel, editor, *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI-01)*, pages 161–166, 2001.
43. V. Haarslev and R. Möller. RACER system description. In R. Goré, A. Leitsch, and T. Nipkov, editors, *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR-01)*, Lecture Notes in Computer Science. Springer, 2001.
44. V. Haarslev and R. 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-04)*, pages 163–173, 2004.
45. V. Haarslev, R. Möller, and A.-Y. Turhan. Exploiting pseudo models for TBox and ABox reasoning in expressive description logics. In R. Goré, A. Leitsch, and T. Nipkov, editors, *Proc. of the International Joint Conference on Automated Reasoning IJCAR'01*, LNAI. Springer Verlag, 2001.
46. M. Hofmann. Proof-theoretic approach to description-logic. In P. Panangaden, editor, *Proc. of the 20th Ann. IEEE Symp. on Logic in Computer Science (LICS-05)*, pages 229–237. IEEE Computer Society Press, 2005.
47. M. Horridge, B. Parsia, and U. Sattler. Laconic and precise justifications in OWL. In *ISWC 08 The International Semantic Web Conference 2008, Karlsruhe, Germany*, 2008.
48. I. Horrocks. *Optimising Tableaux Decision Procedures for Description Logics*. PhD thesis, University of Manchester, 1997.
49. I. Horrocks. Using an expressive description logic: FaCT or fiction? In A. Cohn, L. Schubert, and S. Shapiro, editors, *Proc. of the 6th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-98)*, pages 636–647, 1998.
50. I. Horrocks. Reasoning with expressive description logics: Theory and practice. In A. Voronkov, editor, *Proc. of the 19th Conf. on Automated Deduction (CADE-19)*, number 2392 in Lecture Notes In Artificial Intelligence, pages 1–15. Springer, 2002.

51. I. Horrocks, O. Kutz, and U. Sattler. The even more irresistible $\mathcal{SROIQ}$. In P. Doherty, J. Mylopoulos, and C. Welty, editors, *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-06)*, pages 57–67. AAAI Press, 2006.
52. I. Horrocks and P. Patel-Schneider. Optimizing description logic subsumption. *Journal of Logic and Computation*, 9(3):267–293, 1999.
53. I. Horrocks, P. Patel-Schneider, and F. van Harmelen. From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.
54. I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *Journal of Logic and Computation*, 9(3):385–410, 1999.
55. I. Horrocks and U. Sattler. Optimised reasoning for $\mathcal{SHIQ}$. In *Proc. of the 15th European Conference on Artificial Intelligence*, 2002.
56. I. Horrocks and U. Sattler. A tableaux decision procedure for $\mathcal{SHOIQ}$. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI-05)*. Morgan Kaufmann, Jan. 2005.
57. I. Horrocks and U. Sattler. A tableau decision procedure for $\mathcal{SHOIQ}$. *J. of Automated Reasoning*, 39(3):249–276, 2007.
58. I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for very expressive description logics. *J. of the Interest Group in Pure and Applied Logic*, 8(3):239–264, 2000.
59. U. Hustadt, R. A. Schmidt, and L. Georgieva. A survey of decidable first-order fragments and description logics. *Journal of Relational Methods in Computer Science*, 1:251–276, 2004.
60. A. Kalyanpur, B. Parsia, M. Horridge, and E. Sirin. Finding all justifications of owl dl entailments. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Korea, 2007*, volume 4825 of *Lecture Notes in Computer Science*, pages 267–280, 2007.
61. A. Kalyanpur, B. Parsia, E. Sirin, and B. Cuenca Grau. Repairing unsatisfiable concepts in owl ontologies. In Y. Sure and J. Domingue, editors, *Proc. of the 3rd European Semantic Web Conf. (ESWC'06)*, volume 4011 of *Lecture Notes in Computer Science*, pages 170–184. Springer, 2006.
62. A. Kalyanpur, B. Parsia, E. Sirin, B. Cuenca Grau, and J. Hendler. Swoop: A web ontology editing browser. *J. Web Sem.*, 4(2):144–153, 2006.
63. A. Kalyanpur, B. Parsia, E. Sirin, and J. A. Hendler. Debugging unsatisfiable classes in OWL ontologies. *J. Web Sem.*, 3(4):268–293, 2005.
64. Y. Kazakov. $\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are harder than $\mathcal{SHOIQ}$. In G. Brewka and J. Lang, editors, *Proc. of the 11th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-08)*, pages 274–284. AAAI Press, 2008.
65. M. H. Liffiton and K. A. Sakallah. Algorithms for computing minimal unsatisfiable subsets of constraints. *J. Autom. Reasoning*, 40(1):1–33, 2008.
66. C. Lutz. Complexity of terminological reasoning revisited. In *Proc. of the 6th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR'99)*, Lecture Notes in Computer Science, pages 181–200. Springer, 1999.
67. C. Lutz. The complexity of conjunctive query answering in expressive description logics. In A. Armando, P. Baumgartner, and G. Dowek, editors, *Proc. of the 4th International Joint Conference on Automated Reasoning (IJCAR2008)*, number 5195 in LNAI, pages 179–193. Springer, 2008.
68. C. Lutz, U. Sattler, and F. Wolter. Description logics and the two-variable fragment. In D. McGuiness, P. Pater-Schneider, C. Goble, and R. Möller, editors, *Proc. of the 2001 International Workshop in Description Logics (DL-2001)*, pages 66–75, Stanford, California, USA, 2001.
69. C. Lutz, D. Walther, and F. Wolter. Conservative extensions in expressive description logics. In *Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI-07)*. AAAI Press, 2007.

70. C. Lutz and F. Wolter. Deciding inseparability and conservative extensions in the description logic $\mathcal{EL}$. *Journal of Symbolic Computation*, 45(2):194 – 228, 2010.
71. J. Mendez and B. Suntisrivaraporn. Reintroducing cel as an owl 2 el reasoner. In B. Cuenca Grau, I. Horrocks, B. Motik, and U. Sattler, editors, *Proc. of the 2008 Description Logic Workshop (DL 2009)*, volume 477 of *CEUR-WS*, 2009.
72. T. Meyer, K. Lee, R. Booth, and J. Z. Pan. Finding maximally satisfiable terminologies for the description logic $\mathcal{ALC}$. In *Proc. of the 21st Nat. Conf. on Artificial Intelligence (AAAI'06)*. AAAI Press/The MIT Press, 2006.
73. M. Minsky. A framework for representing knowledge. Technical report, MIT-AI Laboratory, Cambridge, MA, USA, 1974.
74. B. Motik. *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, Universität Karlsruhe, 2006.
75. B. Motik, B. Cuenca Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz. Owl 2 web ontology language profiles. W3C Recommendation, 27 October 2009. `http://www.w3.org/TR/2009/REC-owl2-profiles-20091027/`.
76. B. Motik and U. Sattler. A Comparison of Techniques for Querying Large Description Logic ABoxes. In M. Hermann and A. Voronkov, editors, *Proc. of the 13th Int. Conf. on Logic for Programming Artificial Intelligence and Reasoning (LPAR'06)*, LNCS, Cambodia, 2006. Springer. KAON2 download page: `http://kaon2.semanticweb.org/`.
77. B. Motik, R. Shearer, and I. Horrocks. A hypertableau calculus for $\mathcal{SHIQ}$. In D. Calvanese, E. Franconi, V. Haarslev, D. Lembo, B. Motik, S. Tessaris, and A.-Y. Turhan, editors, *Proc. of the 2007 Description Logic Workshop (DL 2007)*, 2007.
78. B. Motik, R. Shearer, and I. Horrocks. Optimized Reasoning in Description Logics using Hypertableaux. In F. Pfennig, editor, *Proc. of the 23th Conf. on Automated Deduction (CADE-23)*, LNAI, pages 67–83, Germany, 2007. Springer.
79. B. Nebel. Computational complexity of terminological reasoning in BACK. *Artificial Intelligence Journal*, 34(3):371–383, 1988.
80. B. Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence Journal*, 43:235–249, 1990.
81. B. Nebel and K. von Luck. Hybrid reasoning in BACK. In *Proc. of the 3rd Int. Sym. on Methodologies for Intelligent Systems (ISMIS-88)*, pages 260–269. North-Holland Publ. Co., Amsterdam, 1988.
82. M. Ortiz, D. Calvanese, and T. Eiter. Data complexity of query answering in expressive description logics via tableaux. *Journal of Automated Reasoning*, 41(1):61–98, 2008.
83. B. Parsia, E. Sirin, and A. Kalyanpur. Debugging OWL Ontologies. In A. Ellis and T. Hagino, editors, *Proc. of the 14th Int. World Wide Web Conference (WWW2005)*, pages 633–640, Japan, 2005.
84. P. Patel-Schneider, D. L. McGuinness, R. J. Brachman, L. A. Resnick, and A. Borgida. The CLASSIC knowledge representation system: Guiding principles and implementation rational. *SIGART Bulletin*, 2(3):108–113, 1991.
85. M. R. Quillian. Word concepts: A theory and simulation of some basic capabilities. *Behavioral Science*, 12:410–430, 1967. Republished in [24].
86. Racer Systems GmbH & Co. KG. *RacerPro Reference Manual Version 1.9*, Dec. 2005. Available from: `http://www.racer-systems.com/products/racerpro/reference-manual-1-9.pdf`.
87. A. Rector and I. Horrocks. Experience building a large, re-usable medical ontology using a description logic with transitivity and concept inclusions. In *Proc. of the Workshop on Ontological Engineering, AAAI Spring Symposium (AAAI'97)*, 1997.
88. U. Sattler, D. Calvanese, and R. Molitor. Relationships with other formalisms. In *[6]*, pages 137–177. Cambridge University Press, 2003.

89. K. Schild. A correspondence theory for terminological logics: preliminary report. In J. Mylopoulos and R. Reiter, editors, *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI-91)*, Sydney, Australia, 1991.
90. S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In G. Gottlob and T. Walsh, editors, *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI-03)*, pages 355–362, Mexico, 2003. Morgan Kaufmann, Los Altos.
91. M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with unions and complements. Technical Report SR-88-21, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Kaiserslautern, Germany, 1988.
92. M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence Journal*, 48(1):1–26, 1991.
93. E. Sirin and B. Parsia. Pellet system description. In B. Parsia, U. Sattler, and D. Toman, editors, *Description Logics*, volume 189 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006.
94. J. F. Sowa, editor. *Principles of Semantic Networks*. Morgan Kaufmann, Los Altos, 1991.
95. J. F. Sowa. *Encyclopedia of Artificial Intelligence*, chapter Semantic Networks. John Wiley & Sons, New York, 1992.
96. T. Springer and A.-Y. Turhan. Employing description logics in ambient intelligence for modeling and reasoning about complex situations. *Journal of Ambient Intelligence and Smart Environments*, 1(3):235–259, 2009.
97. B. Suntisrivaraporn. *Polynomial-Time Reasoning Support for Design and Maintenance of Large-Scale Biomedical Ontologies*. PhD thesis, Fakultät Informatik, TU Dresden, 2009. http://lat.inf.tu-dresden.de/research/phd/#Sun-PhD-2008.
98. S. Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *Journal of Artificial Intelligence Research*, 12:199–217, May 2000.
99. D. Tsarkov and I. Horrocks. FaCT++ description logic reasoner: System description. In *Proc. of the 3rd Int. Joint Conf. on Automated Reasoning (IJCAR-06)*, 2006. FaCT++ download page: `http://owl.man.ac.uk/factplusplus/`.
100. D. Tsarkov, I. Horrocks, and P. F. Patel-Schneider. Optimising terminological reasoning for expressive description logics. *Journal of Automated Reasoning*, 2007.
101. A.-Y. Turhan. *On the Computation of Common Subsumers in Description Logics*. PhD thesis, TU Dresden, Institute for Theoretical Computer Science, 2007.
102. W3C OWL Working Group. Owl 2 web ontology language document overview. W3C Recommendation, 27th October 2009. `http://www.w3.org/TR/2009/REC-owl2-overview-20091027/`.
103. L. Zhang and S. Malik. Validating sat solvers using an independent resolution-based checker: Practical implementations and other applications. In *Design, Automation and Test in Europe Conference and Exposition (DATE 2003), 3-7 March 2003, Munich, Germany*, pages 10880–10885. IEEE Computer Society, 2003.