A Sequent Calculus for Reasoning in Four-valued Description Logics

Umberto Straccia

Istituto di Elaborazione della Informazione
Via S. Maria, 46-56126 Pisa ITALY
straccia@iei.pi.cnr.it

Abstract. *Description Logics* (DLs, for short) provide a logical reconstruction of the so-called frame-based knowledge representation languages. Originally, four-valued DLs have been proposed in order to develop expressively powerful DLs with tractable subsumption algorithms. Recently, four-valued DLs have been proposed as a model for (multimedia) document retrieval. In this context, the main reasoning task is instance checking. Unfortunately, the known subsumption algorithms for four-valued DLs, based on "structural" subsumption, do not work with respect to the semantics proposed in the DL-based approach to document retrieval. Moreover, they are unsuitable for solving the instance checking problem, as this latter problem is more general than the subsumption problem. We present an alternative decision procedure for four-valued DLs with the aim to solve these problems. The decision procedure is a sequent calculus for instance checking. Since in general the four-valued subsumption problem can be reduced to the four-valued instance checking problem, we obtain a decision procedure for the subsumption problem. Some related complexity results will be presented.

1 Introduction

In the last decade a substantial amount of work has been carried out in the context of *Description Logics* (DLs, for short), which provide a logical reconstruction of the so-called frame-based knowledge representation languages, with the aim of providing a simple well-established denotational semantics to capture the meaning of the most popular features of structured representation of knowledge [20]. *Concepts, roles* and *individuals* are the basic building blocks of these logics. Concepts are expressions which collect the properties, described by means of roles, of a set of individuals. From a logical point of view, concepts can be seen as unary predicates, whereas roles are interpreted as binary predicates. Typically concepts are structured into a hierarchy, induced by the subsumption relation and interpreted as set containment: a concept C subsumes a concept D, iff the set of individuals denoted by the concept D is a subset of the set of individuals denoted by C, *i.e.* in symbols $D \preceq C$. It is a common opinion that subsumption checking is an important reasoning task in DLs. This has motivated a large body of research on the problem of subsumption checking in different DLs [3, 6, 13, 25, 27].

A DL *knowledge base* is a set of *assertions*. An assertion states either that an individual a is an instance of a concept C, written $C(a)$, or that two individuals a and b are related by means of a role R, written $R(a, b)$. The basic inference task with knowledge bases is *instance checking* and amounts to verify whether the individual a is an instance of the concept C with respect to the knowledge base Σ, *i.e.* in symbols $\Sigma \models C(a)$.

Originally, four-valued DLs have been proposed in order to develop expressively powerful DLs with tractable subsumption [21, 22, 23]. It has been shown that the four-valued subsumption relation captures an important subset of two-valued subsumption relationships.

More recently, DLs and their four-valued versions have been proposed in the area of (multimedia) *Document Retrieval* (DR, for short) [12, 17, 18, 19, 26]. In this context, a document base represents a collection of documents described by means of a set of assertions Σ. A query is specified by means of a concept C describing the set of documents to be retrieved. The retrieval of a document identified by the individual d is determined by checking whether $\Sigma \models C(d)$. Four-valued DLs have been proposed as a DR model fulfilling two basic *desiderata* not satisfied by classical DLs: (*i*) since retrieval is defined as the task of retrieving the documents that are *relevant* to the user's information need, the required DL should enforce a notion of entailment in which premises need be *relevant* to conclusions, and to a stronger extent than classical "material" logical implication does; (*ii*) since the *content representation* of a collection of documents cannot be considered as a consistent set of assertions an adequate DR model must be capable of tolerating inconsistencies without loosing its deductive capabilities. Four-valued DLs, which are mainly inspired on [2, 15], satisfy the above points since they (*i*) inherit from Relevance Logic [1] a flavour of "relevance" between premises and conclusion and (*ii*) are paraconsistent and allow *liberal reasoning* [28]. Liberal reasoning has been shown to be suitable for document retrieval purposes [19].

Effectively deciding whether $\Sigma \models C(a)$ requires a calculus. The only known decision procedures in the context of four-valued DLs are subsumption algorithms. These algorithms perform, in an efficient way, "structural" subsumption [3] and are directly inspired on Levesque's algorithm for propositional tautological entailment [15][1]. Unfortunately, they do not work with respect to the semantics proposed in the DL-based DR model. Moreover, they are unsuitable for solving the instance checking problem, as the instance checking problem is a more general problem than the subsumption problem. In fact, in most DLs the subsumption problem can easily reduced to the instance checking problem, as $D \preceq C$ iff $\{D(a)\} \models C(a)$ (where a is an individual) holds[2] and there are cases in which the instance checking problem belongs to a higher complexity

[1] Let α and β be two propositional formulae in conjunctive normal form. Then α tautological entails β iff for each conjunct β_j of β there is a conjunct α_i of α such that $\alpha_i \subseteq \beta_j$, where α_i and β_j are in clausal form. Tautological entailment between two formulae α and β in conjunctive normal form can be verified in time $O(|\alpha||\beta|)$.

[2] See [24] for those cases in which the reduction is not immediate.

class than the corresponding subsumption problem [7].

This paper presents an alternative decision procedure for four-valued DLs with the aim to solve these problems. The decision procedure is a *sequent calculus* for instance checking in four-valued DLs. Since generally the subsumption problem in it can be easily reduced to the instance checking problem, we obtain a decision procedure for subsumption checking for four-valued DLs.

The rest of this paper is organised as follows. In the next section we will briefly recall syntax and two-valued semantics of $\mathcal{ALC}$, a representative DL[3]. In Section 3 we will give to it four-valued semantics, whereas Section 4 highlights some properties of the logic. In Section 5 we will present a calculus for instance checking for four-valued DLs and Section 6 presents some related complexity results. Section 7 concludes.

2 A quick look to Description Logics

The building blocks of the language are primitive concepts (denoted by the letter A), primitive roles (denoted by the letter R) and individuals (denoted by the letter a and b). Complex concepts (C and D) are built out of primitive symbols via the language constructors by means of the syntax rule:

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

For example, the complex concept `Order`$\sqcap \forall$`Sender.CarVendor` is obtained combining the primitive concepts `Order` and `CarVendor` and the primitive role `Sender` by the conjunction ($\sqcap$) and the universal quantification ($\forall$) constructors. An *interpretation* $\mathcal{I}$ consists of a domain $\Delta^{\mathcal{I}}$ (a non-empty set) and of a function mapping a primitive concept into a subset of $\Delta^{\mathcal{I}}$, mapping a primitive role into a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and mapping different individuals into different elements of its domain $\Delta^{\mathcal{I}}$. The interpretation of complex concepts is obtained by appropriately combining the interpretation of their components, *i.e.* $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$, $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$, $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$, $(\forall R.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} : \text{for all } d', \text{ if } (d,d') \in R^{\mathcal{I}} \text{ then } d' \in C^{\mathcal{I}}\}$, and $(\exists R.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} : \text{for some } d', (d,d') \in R^{\mathcal{I}} \text{ and } d' \in C^{\mathcal{I}}\}$.

The subsumption relation between two concepts is defined in terms of set containment: a concept C *subsumes* a concept D, written $D \preceq_2 C$, iff for all interpretations $\mathcal{I}$, $D^{\mathcal{I}} \subseteq C^{\mathcal{I}}$.

An assertion is of type $C(a)$ or of type $R(a,b)$, where C, R and a, b are a concept, a role and two individuals, respectively. A *primitive assertion* (resp. *negated primitive assertion*) is an assertion of type $A(a)$ (resp. $\neg A(a)$). An interpretation $\mathcal{I}$ satisfies an assertion $C(a)$ just in case $a^{\mathcal{I}} \in C^{\mathcal{I}}$, whereas $\mathcal{I}$ satisfies an assertion $R(a,b)$ whenever $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$. Given a set of assertions Σ, a concept C and an individual a, then Σ *entails* $C(a)$, written $\Sigma \models_2 C(a)$, iff every interpretation satisfying all the assertions in Σ also satisfies $C(a)$. It is easily

[3] Although we restrict our attention to $\mathcal{ALC}$, our framework can be applied to other DLs as well.

verified that in $\mathcal{ALC}$, $D \preceq_2 C$ iff $\{D(a)\} \models_2 C(a)$ (where a is an individual) holds. Hence, subsumption can be reduced to instance checking in $\mathcal{ALC}$. This property holds for most DLs presented in the literature and in particular it will hold for all DLs we will consider in this paper.

3 Four-valued Description Logics

The four-valued semantics for $\mathcal{ALC}$ is based on a four-valued semantics similar as in [23], but we will also present a different choice for the semantics of the $(\forall)$ constructor.

As usual, the four truth values are the elements of $2^{\{t,f\}}$, the powerset of $\{t,f\}$, *i.e.* $\{t,f\}$, $\{\}$, $\{t\}$ and $\{f\}$. These values are known as *contradiction*, *unknown*, *true* and *false*, respectively. An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non empty set $\Delta^{\mathcal{I}}$ (the *domain* of $\mathcal{I}$) and a function $\cdot^{\mathcal{I}}$ (the *interpretation function* of $\mathcal{I}$) such that (i) $\cdot^{\mathcal{I}}$ maps every concept into a function from $\Delta^{\mathcal{I}}$ to $2^{\{t,f\}}$; (ii) $\cdot^{\mathcal{I}}$ maps every role into a function from $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ to $2^{\{t,f\}}$; (iii) $\cdot^{\mathcal{I}}$ maps every individual into an element of $\Delta^{\mathcal{I}}$ and (iv) $a^{\mathcal{I}} \neq b^{\mathcal{I}}$, if $a \neq b$. Given an interpretation $\mathcal{I}$, the *positive extension* of a concept C, written $[\![C^{\mathcal{I}}]\!]^+$, is defined as the set $\{d \in \Delta^{\mathcal{I}} : t \in C^{\mathcal{I}}(d)\}$, whereas the *negative extension* of a concept C, written $[\![C^{\mathcal{I}}]\!]^-$, is defined as the set $\{d \in \Delta^{\mathcal{I}} : f \in C^{\mathcal{I}}(d)\}$. The positive and negative extension of a role is defined similarly. It is easily verified that a two-valued interpretation is an interpretation $\mathcal{I}$ such that for every concept C, $[\![C^{\mathcal{I}}]\!]^- = \Delta^{\mathcal{I}} \setminus [\![C^{\mathcal{I}}]\!]^+$ and for all roles R, $[\![R^{\mathcal{I}}]\!]^- = \Delta^{\mathcal{I}} \setminus [\![R^{\mathcal{I}}]\!]^+$. Unlike two-valued semantics, the positive extension and the negative extension need not to be complement of each other.

The extensions of concepts have to meet certain restrictions, designed so that the formal semantics respects the informal meaning of concepts and roles. They are: $[\![(C \sqcap D)^{\mathcal{I}}]\!]^+ = [\![C^{\mathcal{I}}]\!]^+ \cap [\![D^{\mathcal{I}}]\!]^+$, $[\![(C \sqcap D)^{\mathcal{I}}]\!]^- = [\![C^{\mathcal{I}}]\!]^- \cup [\![D^{\mathcal{I}}]\!]^-$, $[\![(C \sqcup D)^{\mathcal{I}}]\!]^+ = [\![C^{\mathcal{I}}]\!]^+ \cup [\![D^{\mathcal{I}}]\!]^+$, $[\![(C \sqcup D)^{\mathcal{I}}]\!]^- = [\![C^{\mathcal{I}}]\!]^- \cap [\![D^{\mathcal{I}}]\!]^-$, $[\![(\neg C)^{\mathcal{I}}]\!]^+ = [\![C^{\mathcal{I}}]\!]^-$, $[\![(\neg C)^{\mathcal{I}}]\!]^- = [\![C^{\mathcal{I}}]\!]^+$, $[\![(\exists R.C)^{\mathcal{I}}]\!]^+ = [\![(\neg \forall R.\neg C)^{\mathcal{I}}]\!]^+$, $[\![(\exists R.C)^{\mathcal{I}}]\!]^- = [\![(\neg \forall R.\neg C)^{\mathcal{I}}]\!]^-$. It is worth noting that the semantics for the $(\exists)$ constructor is given in terms of the $(\forall)$ constructor. For it, we present two different semantics:

Type A: for each $d, d' \in \Delta^{\mathcal{I}}$

$$t \in (\forall R.C)^{\mathcal{I}}(d) \text{ iff } \forall\, e \in \Delta^{\mathcal{I}}, t \in R^{\mathcal{I}}(d,e) \text{ implies } t \in C^{\mathcal{I}}(e)$$
$$f \in (\forall R.C)^{\mathcal{I}}(d) \text{ iff } \exists\, e \in \Delta^{\mathcal{I}}, t \in R^{\mathcal{I}}(d,e) \text{ and } f \in C^{\mathcal{I}}(e)$$

Type B: for each $d, d' \in \Delta^{\mathcal{I}}$

$$t \in (\forall R.C)^{\mathcal{I}}(d) \text{ iff } \forall\, e \in \Delta^{\mathcal{I}}, f \in R^{\mathcal{I}}(d,e) \text{ or } t \in C^{\mathcal{I}}(e)$$
$$f \in (\forall R.C)^{\mathcal{I}}(d) \text{ iff } \exists\, e \in \Delta^{\mathcal{I}}, t \in R^{\mathcal{I}}(d,e) \text{ and } f \in C^{\mathcal{I}}(e)$$

Notice that type B semantics is used in [23], whereas type A semantics is used in [19].

In the following, the default will be the semantics of type A. A concept C is *equivalent* to a concept D, written $C \equiv_4^A D$, iff $[\![C^{\mathcal{I}}]\!]^+ = [\![D^{\mathcal{I}}]\!]^+$, for every interpretation $\mathcal{I}$. A concept C *subsumes* a concept D, written $D \preceq_4^A C$, iff $[\![D^{\mathcal{I}}]\!]^+ \subseteq [\![C^{\mathcal{I}}]\!]^+$, for every interpretation $\mathcal{I}$. A concept C is *coherent* iff there is an interpretation $\mathcal{I}$ such that $[\![C^{\mathcal{I}}]\!]^+ \neq \emptyset$.

With respect to assertions, we have the following definitions. An interpretation $\mathcal{I}$ *satisfies an assertion* $C(a)$, iff $t \in C^{\mathcal{I}}(a^{\mathcal{I}})$, whereas $\mathcal{I}$ *satisfies an assertion* $R(a,b)$ iff $t \in R^{\mathcal{I}}(a^{\mathcal{I}}, b^{\mathcal{I}})$. Given two $\mathcal{ALC}$ assertions α and β, α is *equivalent* to β, written $\alpha \equiv_4^A \beta$, iff for every interpretation $\mathcal{I}$, $\mathcal{I}$ satisfies α iff $\mathcal{I}$ satisfies β. An interpretation $\mathcal{I}$ *satisfies* (is a *model* of) a knowledge base Σ iff $\mathcal{I}$ satisfies all assertions in Σ. A knowledge base Σ *entails* an assertion $C(a)$, written $\Sigma \models_4^A C(a)$, iff all models of Σ satisfy $C(a)$. Finally, all the above definitions are given for the case of two-valued semantics and type B semantics too. In these cases we will use $\cdot_2$ and $\cdot_4^B$, respectively, in place of $\cdot_4^A$. For instance, we will write $\Sigma \models_4^B C(a)$, if Σ entails $C(a)$ with respect to type B semantics and we will write $C \equiv_2 D$ iff C and D are equivalent with respect to two-valued semantics.

4 Some properties of the semantics

We will not go into a detailed discussion since a detailed one can be found in [19].

Just note that in our four-valued logic, every concept is coherent, whereas this is not true with respect to two-valued semantics, *e.g.* $A \sqcap \neg A$ is not a "two-valued" coherent concept. Similarly, every knowledge base is satisfiable.

As in two-valued $\mathcal{ALC}$, it is easily verified that $D \preceq_4^A C$ iff $\{D(a)\} \models_4^A C(a)$, and $D \preceq_4^B C$ iff $\{D(a)\} \models_4^B C(a)$, where a is an individual. Therefore, in four-valued $\mathcal{ALC}$ the subsumption problem can be reduced to the instance checking problem. This property holds for all four-valued DLs we will analyse in this paper.

Reasoning in our logic is *sound* with respect to two-valued semantics. In fact, it is easily verified that for all knowledge bases Σ, for all assertions α and for all concepts C and D, $D \preceq_4^A C$ implies $D \preceq_2 C$ and $\Sigma \models_4^A \alpha$ implies $\Sigma \models_2 \alpha$. From $A \sqcap \neg A \preceq_2 B$ and $A \sqcap \neg A \not\preceq_4^A B$, it follows that $\preceq_4^A \subset \preceq_2$ and $\models_4^A \subset \models_2$ hold.

In [19, 23] it has already shown that $\preceq_4^B$ and $\models_4^A$ capture an *interesting* subset of $\preceq_2$ and $\models_2$, respectively. In what follows we will show which inferences, licensed in two-valued semantics, are left out by our semantics. It is quite obvious that the so-called paradoxes of logical implication, *i.e.* $\{C(a), \neg C(a)\} \models_2 D(b)$ and $\emptyset \models_2 (C \sqcup \neg C)(a)$ do not hold in our type A and type B semantics. Generally, modus ponens is not a valid inference rule in our logic, *i.e.* $\{(C \sqcap (\neg C \sqcup D))(a)\} \not\models_4^A D(a)$. But, the semantics of type A allows a restricted form of modus ponens, called *modus ponens on roles*: for all concepts C and D, for any role R, and for all individuals a, b, $\{(\forall R.C)(a), R(a,b)\} \models_4^A C(b)$ and $(\exists R.C) \sqcap (\forall R.D) \preceq_4^A \exists R.(C \sqcap D)$. This kind of inference is not allowed in type B semantics. Hence, $\models_4^B \neq \models_4^A$ and $\preceq_4^B \neq \preceq_4^A$ hold. In the next section we will see that type B semantics is weaker than type A semantics, by showing

that $\models_4^B \subseteq \models_4^A$. Thus, $\preceq_4^B \subset \preceq_4^A$ and $\models_4^B \subset \models_4^A$ hold. Unfortunately, the above additional inference has a cost in terms of computational complexity. In fact, as we will see in Section 6, checking whether $\Sigma \models_4^A C(a)$ is harder than checking whether $\Sigma \models_4^B C(a)$.

Reasoning by cases does not work within our type A and type B semantics. Consider the following knowledge base $\Sigma' = \{\mathtt{P(p)}, \mathtt{HS(p,c1)}, \mathtt{HS(p,c2)}, \mathtt{F(c1,c2)}, \mathtt{F(c2,c3)}, \mathtt{I(c1)}, \neg\mathtt{I(c3)}\}$, where `P`, `HS`, `F` and `I` stand for `Person`, `HasStudent`, `Friend` and `Ill`, respectively. Consider now the assertion $\alpha = (\exists \mathtt{HS}.(\mathtt{I} \sqcap \exists \mathtt{F}.\neg\mathtt{I}))(\mathtt{p})$. It can be verified that $\Sigma' \not\models_4^A \alpha$ and $\Sigma' \models_2 \alpha$ hold. In fact, $\Sigma' \models_2 \alpha$ since in each two-valued interpretation $\mathcal{I}$ satisfying Σ', either $\mathtt{I(c2)}$ is true or $\mathtt{I(c2)}$ is false. On the other hand, $\Sigma' \not\models_4^A \alpha$ holds since it could be the case $\mathtt{I}^{\mathcal{I}}(\mathtt{c2}^{\mathcal{I}}) = \emptyset$. That is, we are uncertain about `c2`'s illness: we have no "relevant" information about `c2`'s illness. In [19] it has been shown that $\Sigma' \not\models_4^A \alpha$ is considered suitable for document retrieval purposes.

It is worth noting that there is no "top" concept within type A and type B semantics, *i.e.* there is no concept C such that $\emptyset \models_4^A C(a)$, for all individuals a. This plays an important role in the example above. In fact, suppose that we would admit a "top" concept $\top$ with semantics $[\![\top^{\mathcal{I}}]\!]^+ = \Delta^{\mathcal{I}}$ and $[\![\top^{\mathcal{I}}]\!]^- = \emptyset$. If we replace in Σ' and α above, `I` and $\neg$`I` with $\exists$`R`.$\top$ and $\forall$`R.A`, respectively, then it can be verified that $\Sigma'' \models_4^A \alpha'$ and $\Sigma'' \not\models_4^B \alpha'$ hold, where Σ'' and α' are the result of the substitutions. This is due to the fact that $\Sigma'' \models_4^A \alpha'$ relies on the relations $\mathtt{I} \sqcup \neg\mathtt{I} \equiv_2 \top$ and $\exists\mathtt{R}.\top \sqcup \forall\mathtt{R.A} \equiv_4^A \top$, whereas $\exists\mathtt{R}.\top \sqcup \forall\mathtt{R.A} \not\equiv_4^B \top$. As we will see in Section 6, admitting a top concept changes the computational complexity of instance checking.

5 A sequent calculus for entailment

In the following, we will assume type A semantics as the default semantics.

Effectively deciding whether, given a knowledge base Σ and an assertion α, $\Sigma \models_4^A \alpha$, requires a calculus. The one we have developed is a *sequent calculus* [9, 11].

The main idea behind our calculus for entailment is that in order to prove $\Sigma \models_4^A \alpha$ we try to prove the "sequent" $\Sigma \rightarrow \alpha$. Without loss of generality we can restrict our attention to assertions in *negation normal form* (NNF), *i.e.* the negation constructor can be only in front of primitive symbols[4]. Furthermore, we add to the alphabets of concepts, roles and individuals another alphabet of symbols, called *variables* (denoted by x and y). The alphabet of *objects* is the union of the alphabets of variables and individuals (objects are denoted by v and w). An interpretation $\mathcal{I}$ is extended to variables by mapping them into an element of its domain $\Delta^{\mathcal{I}}$.

A *sequent* is an expression of the form $\Gamma \rightarrow \Delta$, where $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$ and $\Delta = \{\delta_1, \ldots, \delta_m\}$ are finite sets of assertions, with $n + m \geq 1$. Γ is called the

[4] It is easy to see that every assertion can be transformed in linear time and space into an equivalent assertion in NNF, respectively. Note that in $\mathcal{ALC}$ roles are already in NNF.

antecedent and Δ is called the *consequent*. A sequent $\{\gamma_1, \ldots, \gamma_n\} \rightarrow \{\delta_1, \ldots, \delta_m\}$ is *satisfiable* iff there is an interpretation $\mathcal{I}$ such that if $\mathcal{I}$ satisfies *all* assertions in $\{\gamma_1, \ldots, \gamma_n\}$ then $\mathcal{I}$ satisfies *some* assertion in $\{\delta_1, \ldots, \delta_m\}$. A sequent $\Gamma \rightarrow \Delta$ is *valid* iff all interpretations satisfy $\Gamma \rightarrow \Delta$. A sequent $\Gamma \rightarrow \Delta$ is *falsifiable* iff it is not valid. Just note that $\Sigma \models_4^A \alpha$ iff the sequent $\Sigma \rightarrow \alpha$ is valid. For ease of notation we will often omit braces and operations of set-theoretic union, thus writing *e.g.* γ, Δ, Γ in place of $\{\gamma\} \cup \Delta \cup \Gamma$.

An *axiom* (*of type* AX) is a sequent of the form $\alpha, \Gamma \rightarrow \Delta, \alpha$. From the definition it is immediate to see that all axioms are valid sequents. A sequent calculus is based on a number of *rules of inference* operating on sequents. Rules fall naturally into two categories: those operating on assertions occurring in the antecedent, and those operating on assertions occurring in the consequent. Every rule consists of one or two "upper" sequents called *premises* and of a "lower" sequent called *conclusion*. The rules of the calculus for entailment with respect to type A semantics are defined as follows:

$$(\sqcap \rightarrow) \ \frac{C(v), D(v), \Gamma \rightarrow \Delta}{(C \sqcap D)(v), \Gamma \rightarrow \Delta} \qquad (\rightarrow \sqcap) \ \frac{\Gamma \rightarrow \Delta, C(v) \ \Gamma \rightarrow \Delta, D(v)}{\Gamma \rightarrow \Delta, (C \sqcap D)(v)}$$

$$(\sqcup \rightarrow) \ \frac{C(v), \Gamma \rightarrow \Delta \ D(v), \Gamma \rightarrow \Delta}{(C \sqcup D)(v), \Gamma \rightarrow \Delta} \qquad (\rightarrow \sqcup) \ \frac{\Gamma \rightarrow \Delta, C(v), D(v)}{\Gamma \rightarrow \Delta, (C \sqcup D)(v)}$$

$$(\forall \rightarrow) \ \frac{(\forall R.C)(v), R(v,w), C(w), \Gamma \rightarrow \Delta}{(\forall R.C)(v), R(v,w), \Gamma \rightarrow \Delta} \qquad (\rightarrow \forall) \ \frac{R(v,x), \Gamma \rightarrow \Delta, D(x)}{\Gamma \rightarrow \Delta, (\forall R.D)(v)}$$

$$(\exists \rightarrow) \ \frac{R(v,x), C(x), \Gamma \rightarrow \Delta}{(\exists R.C)(v), \Gamma \rightarrow \Delta} \qquad (\rightarrow \exists) \ \frac{R(v,w), \Gamma \rightarrow \Delta, (\exists R.C)(v), C(w)}{R(v,w), \Gamma \rightarrow \Delta, (\exists R.C)(v)}$$

where x is a new variable (called also *eigenvariable*) and v, w are objects. Of course, in order to prevent infinite applications, the $(\forall \rightarrow)$ rule can be applied only if $C(w) \notin \Gamma$. Similarly for the $(\rightarrow \exists)$ rule.

A deduction can easily be represented as a tree (growing upwards): a *deduction tree* is a tree whose nodes are each labelled with a sequent and in which a sequent labelling a node may be obtained through the application of a rule of inference to the sequents labelling their children nodes. The sequent labelling the root of a deduction tree is called *conclusion* of the deduction tree[5]. A *proof tree* is a deduction tree whose leaves are labelled with an axiom. A sequent $\Gamma \rightarrow \Delta$ is *provable*, written $\Gamma \mid\sim_4^A \Delta$, iff there is a proof tree of which it is the conclusion. We will write $\Gamma \mid\sim_4^B \Delta$ and $\Gamma \mid\sim_2 \Delta$, whenever the calculus refers to axioms and rules for type B semantics and two-valued semantics, respectively. A proof of a sequent $\Gamma \rightarrow \Delta$ proceeds backward, by constructing a proof tree with root $\Gamma \rightarrow \Delta$ and applying the rules until each branch reaches an axiom. For example, the proof of the valid sequent $(C \sqcup D)(a), E(a) \rightarrow (E \sqcap (C \sqcup D))(a)$ starts with a root node labelled $(C \sqcup D)(a), E(a) \rightarrow (E \sqcap (C \sqcup D))(a)$, applies the $(\sqcup \rightarrow)$ rule to it, generates two new nodes labelled with the sequents

[5] Trees are represented reversed, *i.e.* the root is at the bottom.

$C(a), E(a) \rightarrow (E \sqcap (C \sqcup D))(a)$ and $D(a), E(a) \rightarrow (E \sqcap (C \sqcup D))(a)$ and proceeds backward until each branch reaches an axiom. The deduction tree is shown below.

$$\dfrac{\dfrac{\dfrac{}{C(a), E(a) \rightarrow E(a)} \quad \dfrac{\dfrac{}{C(a), E(a) \rightarrow C(a), D(a)}}{C(a), E(a) \rightarrow (C \sqcup D)(a)}}{C(a), E(a) \rightarrow (E \sqcap (C \sqcup D))(a)} \quad \dfrac{\dfrac{\dfrac{}{D(a), E(a) \rightarrow E(a)} \quad \dfrac{\dfrac{}{D(a), E(a) \rightarrow C(a), D(a)}}{D(a), E(a) \rightarrow (C \sqcup D)(a)}}{D(a), E(a) \rightarrow (E \sqcap (C \sqcup D))(a)}}{(C \sqcup D)(a), E(a) \rightarrow (E \sqcap (C \sqcup D))(a)}$$

5.1 Provability, Soundness and Completeness

As first, it can be verified that any deduction tree is finite. Therefore, the deduction of a sequent terminates after a finite number of rule applications. Soundness of the calculus, *i.e.* if the sequent $\Gamma \rightarrow \Delta$ is provable then $\Gamma \rightarrow \Delta$ is valid, can easily be proven by observing that every axiom is valid and that for each of the rules, the conclusion of a rule is valid iff all premises of the rule are valid.

Completeness is proved by means of Hintikka sets (sets of signed assertions) [8, 9]. Let T and NT be two new symbols not appearing in any considered alphabet. *Signed assertions of type-a*, *type-b*, *type-c* and *type-d* and their *components* are defined as follows (C, D are concepts, R is a role and v, w are objects):

Type-a assertion	Components		Type-b assertion	Components	
α^a	α_1^a	α_2^a	α^b	α_1^b	α_2^b
$T(C \sqcap D)(v)$	$TC(v)$	$TD(v)$	$T(C \sqcup D)(v)$	$TC(v)$	$TD(v)$
$NT(C \sqcup D)(v)$	$NTC(v)$	$NTD(v)$	$NT(C \sqcap D)(v)$	$NTC(v)$	$NTD(v)$
Type-c assertion	Components		Type-d assertion	Components	
α^c	α_1^c	α_2^c	α^d	α_1^d	α_2^d
$T(\forall R.C)(v)$	$TR(v,w)$	$TC(w)$	$T(\exists R.C)(v)$	$TR(v,w)$	$TC(w)$
$NT(\exists R.C)(v)$	$TR(v,w)$	$NTC(w)$	$NT(\forall R.C)(v)$	$TR(v,w)$	$NTC(w)$

Let α be an assertion. Then $T\alpha$ and $NT\alpha$ are called *conjugated signed assertions*. We define then satisfaction of signed assertions as follows. Let $\mathcal{I}$ be an interpretation and α an assertion. Then $\mathcal{I}$ *satisfies* $T\alpha$ iff $\mathcal{I}$ satisfies α, whereas $\mathcal{I}$ *satisfies* $NT\alpha$ iff $\mathcal{I}$ does not satisfy α.

Let H be a set of objects. A *Hintikka set* S *with respect to* H is a set of signed assertions such that the following conditions hold for all signed assertions α^a, α^b, α^c, α^d of type a, b, c, d, respectively: (i) no conjugated signed primitive assertions or conjugated signed negated primitive assertions are in S; (ii) if a type-a assertion α^a is in S, then both α_1^a and α_2^a are in S; (iii) if a type-b assertion α^b is in S, then either α_1^b is in S or α_2^b is in S; (iv) if a type-c assertion α^c is in S, then for every object $w \in H$, if α_1^c is in S then α_2^c is in S; (v) if a type-d assertion α^d is in S, then there is at least one object $w \in H$ such that both α_1^d and α_2^d are in S. It can easily be shown that

Lemma 1. *Every Hintikka set S w.r.t. a set of objects H is satisfiable in an interpretation with domain H.* $\dashv$

Theorem 2. *A sequent $\Gamma \rightarrow \Delta$ is valid w.r.t. type A semantics iff $\Gamma \mid\sim_4^A \Delta$, where $\mid\sim_4^A$ is determined by axioms of type AX and the rules $(\sqcap \rightarrow)$, $(\rightarrow \sqcap)$, $(\sqcup \rightarrow)$, $(\rightarrow \sqcup)$, $(\forall \rightarrow)$, $(\rightarrow \forall)$, $(\exists \rightarrow)$ and $(\rightarrow \exists)$.* $\dashv$

Proof. If there is a proof tree for $\Gamma \rightarrow \Delta$ then, from the correctness of the rules we have that $\Gamma \rightarrow \Delta$ is valid. Otherwise, pick up a deduction tree not being a proof tree, and which cannot be expanded any more. Therefore, there is a path from the conclusion to a non axiom leaf of the tree. Let LHS be the union of all assertions occurring in the left hand side of each sequent along that path and RHS be the union of all assertions occurring in the right hand side of any such sequent. Let $S = \{T\alpha : \alpha \in LHS\} \cup \{NT\alpha : \alpha \in RHS\}$. Let H be the set of objects appearing in S. It can be proven that S is a Hintikka set. From Lemma 1 it follows that S is satisfiable with respect to H. Since $\Gamma \subseteq LHS$ and $\Delta \subseteq RHS$, it follows that $\Gamma \rightarrow \Delta$ is falsifiable. □

With respect to type B semantics, the rules for the $(\forall)$ constructor no longer preserve validity between conclusion and premise. We can easily modify these rules in order to obtain completeness with respect to type B semantics. In fact, let $(\forall_s)$ be the rule (x is a new variable)

$$(\forall_s)\ \frac{(\forall R.C)(v), C(x), \Gamma \rightarrow \Delta, (\forall R.D)(v), D(x)}{(\forall R.C)(v), \Gamma \rightarrow \Delta, (\forall R.D)(v)}$$

It is easy to see that these rules reflects the type B semantics of the $(\forall)$ constructor in the sense that a conclusion is valid iff its premise is valid. If we consider the axioms and rules for type A semantics where the rules $(\forall \rightarrow)$ and $(\rightarrow \forall)$ are replaced with the rule $(\forall_s)$, then, similarly as for Theorem 2, it can be shown that

Theorem 3. *A sequent $\Gamma \rightarrow \Delta$ is valid w.r.t. type B semantics iff $\Gamma \mid\sim_4^B \Delta$, where $\mid\sim_4^B$ is determined by axioms of type AX and the rules $(\sqcap \rightarrow)$, $(\rightarrow \sqcap)$, $(\sqcup \rightarrow)$, $(\rightarrow \sqcup)$, $(\forall_s)$, $(\exists \rightarrow)$ and $(\rightarrow \exists)$.* $\dashv$

Since the $(\forall_s)$ rule is a special case of the $(\rightarrow \forall)$ rule, it follows that $\Gamma \mid\sim_4^B \Delta$ implies $\Gamma \mid\sim_4^A \Delta$. Therefore, $\models_4^B \subseteq \models_4^A$ (strictness of the inclusion follows from Section 4).

Completeness with respect to standard two-valued $\mathcal{ALC}$ is obtained by simply adding the following rules to our calculus:

$$(\neg \rightarrow)\ \frac{\Gamma \rightarrow \Delta, A(v)}{\neg A(v), \Gamma \rightarrow \Delta} \qquad (\rightarrow \neg)\ \frac{\Gamma \rightarrow \Delta, \neg A(v)}{A(v), \Gamma \rightarrow \Delta}$$

where A is a primitive concept. It is easily verified that for both rules the conclusion is valid iff its premise is valid. Hence, validity is preserved.

Theorem 4. *A sequent $\Gamma \rightarrow \Delta$ is valid w.r.t. two-valued semantics iff $\Gamma \mid\sim_2 \Delta$, where $\mid\sim_2$ is determined by axioms of type AX and the rules $(\sqcap \rightarrow)$, $(\rightarrow \sqcap)$, $(\sqcup \rightarrow)$, $(\rightarrow \sqcup)$, $(\forall \rightarrow)$, $(\rightarrow \forall)$, $(\exists \rightarrow)$, $(\rightarrow \exists)$, $(\neg \rightarrow)$ and $(\rightarrow \neg)$.* ⊣

It should be noted that another complete calculus with respect to two-valued semantics is obtained by considering axioms (*of type* AX') of the form $\neg A(v), A(v), \Gamma \rightarrow \Delta$ and only the left hand side rules. The following theorem can easily be shown.

Theorem 5. *Let $\mid\sim_2'$ be the derivation relation determined by axioms of type AX' and the rules $(\sqcap \rightarrow)$, $(\sqcup \rightarrow)$, $(\forall \rightarrow)$ and $(\exists \rightarrow)$. Then*

1. *if C and D are two concepts, then $C \preceq_2 D$ iff $C \sqcap \neg D$ is not coherent;*
2. *a concept C is coherent iff the knowledge base $\{C(a)\}$ is satisfiable, where a is an individual;*
3. *a knowledge base Σ is satisfiable iff the sequent $\Sigma \rightarrow \emptyset$ is not valid;*
4. *a sequent $\Gamma \rightarrow \emptyset$ is valid iff $\Gamma \mid\sim_2' \emptyset$.* ⊣

Therefore, the above theorem establishes that the well known (refutation based) constraint propagation method [25] devised for reasoning in two-valued DLs is a special case within our framework, *i.e.* the decision procedure based on $\mid\sim_2'$ is *exactly the same* decision procedure as the standard constraint propagation method for two-valued knowledge base satisfiability checking.

6 Computational complexity

In the following, $\mathcal{AL}$ is the DL with syntax rule[6]

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

$\mathcal{ALE}$ is $\mathcal{AL}$ plus qualified existential quantification $\exists R.C$. $\mathcal{ALE}_1^-$ is the DL with syntax rule

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

and $\mathcal{ALE}_2^-$ is $\mathcal{ALE}_1^-$ plus unqualified existential quantification $\exists R$.

It is well known that the (in)coherence problem is PSPACE-complete for two-valued $\mathcal{ALC}$ [25]. By adopting the same reduction technique as for two-valued $\mathcal{ALC}$, the following theorem can be shown.

Theorem 6. *The instance checking problem and the subsumption checking problem w.r.t. type A semantics are PSPACE-complete for $\mathcal{ALC}$.* ⊣

[6] The 2-valued extension of $\exists R$ is $\{d \in \Delta^{\mathcal{I}} : \exists d' \in \Delta^{\mathcal{I}}$ such that $(d, d') \in R^{\mathcal{I}}\}$. We use $\bot$ as a macro for $\neg\top$.

Proof. We recall that the PSPACE-hardness of the (two-valued) coherence problem has been shown by means of a reduction of the validity problem for quantified boolean formulae [25], which is PSPACE-complete [10]. As in [25], let $P.M$ be a quantified boolean formula, where P (the prefix) is a sequence of quantification over boolean variables and M (the matrix) is a set of clauses (a clause is a set of boolean variables). For instance, $\forall x \exists y.(x \vee \neg y) \wedge \neg y$ is a valid boolean formula, where $P = \forall x \exists y$ and $M = (x \vee \neg y) \wedge \neg y$[7]. Consider the reduction of $P.M$ into the $\mathcal{ALC}$ concept $[P.M] = [P]^0 \sqcap [C_1]^0 \sqcap \ldots \sqcap [C_n]^0$, as in [25]. It has been shown that $P.M$ is a valid boolean formula iff $[P.M]$ is coherent. Consider $[P]^0 \sqcap \neg D$, where $D = D_1 \sqcup \ldots \sqcup D_1$ and D_i is the NNF of $\neg[C_i]^0$. It follows that $[P.M]$ is incoherent iff $[P] \sqcap \neg D$ is incoherent iff $\{[P](a)\} \models_2 D(a)$, where a is an individual. Therefore, by completeness it follows that $\{[P](a)\} \models_2 D(a)$ iff $[P](a) \mid\sim_2 D(a)$. It can be verified that any deduction $[P](a) \mid\sim_2 D(a)$ does not rely on the rules $(\neg \rightarrow)$ and $(\rightarrow \neg)$. Hence, $[P](a) \mid\sim_2 D(a)$ iff $[P](a) \mid\sim_4^A D(a)$. Therefore, $[P.M]$ is incoherent iff $\{[P](a)\} \models_4^A D(a)$. As a consequence, instance checking is a PSPACE-hard problem. Analogously, $[P.M]$ is incoherent iff $[P] \preceq_4^A D$. Hence, subsumption checking is a PSPACE-hard problem too.

Furthermore, checking whether $\Sigma \mid\sim_4^A C(a)$ can be done in polynomial space. This is obtained by adopting a *trace rule* in place of the $(\exists \rightarrow)$ rule which is identical to the trace rule for the $(\exists)$ constructor in the case of the two-valued constraint propagation technique [25]. Therefore, instance checking is a PSPACE-complete problem with respect to type A semantics. Since subsumption can be reduced to instance checking, it follows that the subsumption problem is PSPACE-complete too. □

This result shows that modus ponens on roles is effectively sufficient to get PSPACE-hardness. Moreover, from the proof it follows that the theorem holds for $\mathcal{ALC}$ without the negation constructor $(\neg)$ too.

By using the same reduction as described in [7], where PSPACE-completeness of the instance checking problem with respect to two-valued $\mathcal{ALE}$ is shown, it can be verified that Theorem 6 holds for the language $\mathcal{ALE}_2^-$ too.

Theorem 7. *The instance checking problem w.r.t. type A semantics is PSPACE-complete for* $\mathcal{ALE}_2^-$. ⊣

Similarly, by proceeding as in [5], where NP-completeness of the two-valued subsumption problem is shown for $\mathcal{ALE}$ and $\mathcal{FLE}^-$, it can be proven that

Theorem 8. *Subsumption checking w.r.t. type A semantics is a NP-complete problem for* $\mathcal{ALE}_2^-$. ⊣

Hence, with respect to type A semantics instance checking is strictly more difficult than subsumption for $\mathcal{ALE}_2^-$.

By using type B semantics, modus ponens on roles is not a valid inference rule. As a consequence we can prove that checking the falsifiability of sequents

[7] For readability, we write $(x \vee \neg y) \wedge \neg y$ in place of $\{\{x, \neg y\}, \{\neg y\}\}$.

can be done in non-deterministic polynomial time. Since propositional logic is a sub language of $\mathcal{ALC}$ and since it is well known that propositional tautological entailment is a coNP-complete problem, it follows that:

Theorem 9. *The instance checking problem and the subsumption problem w.r.t. type B semantics are coNP-complete for $\mathcal{ALC}$.* ⊣

As noted in Section 4, type B semantics has a weaker entailment relation than type A semantics. On the other hand, from a computational point of view the computational complexity switches from coNP to PSPACE.

Since in certain circumstances reasoning by cases does not hold, the instance checking problem can be even in P. Suppose that Σ is formed out by $\mathcal{AL}$ assertions, and consider an assertion $C(a)$ which is an $\mathcal{ALE}_1^-$ assertion.

As shown in [7], the instance checking problem with respect to two-valued semantics is coNP-hard in this case. Whereas, it can be shown that

Theorem 10. *Let Σ be formed out by $\mathcal{AL}$ assertions and let C be an $\mathcal{ALE}_1^-$ concept. Then checking whether $\Sigma \models_4^A C(a)$ and checking whether $\Sigma \models_4^B C(a)$ can be done in polynomial time.* ⊣

Proof Sketch. The proof is based on the fact that the $(\rightarrow \exists)$ rule can be replaced by the rule

$$(\rightarrow_T \exists) \quad \frac{R(v,w), \Gamma \rightarrow \Delta, C(w)}{R(v,w), \Gamma \rightarrow \Delta, (\exists R.C)(v)}$$

and the $(\forall_s)$ rule (in the case of type B semantics) can be replaced by the rule

$$(\forall_{sT}) \quad \frac{C(x), \Gamma \rightarrow \Delta, D(x)}{(\forall R.C)(v), \Gamma \rightarrow \Delta, (\forall R.D)(v)}$$

and observing that in all rules Δ is always empty. Therefore, the number and the dimension of the leaves of a deduction tree, and the number of deduction trees to be generated are bounded by the dimension of the root node. □

It is worth noting that, from [7] it follows that whenever unqualified existential concepts of type $\exists R$ are allowed to occur in C, the theorem does not hold for type A semantics (again, we rely on the relation $\exists R \sqcup \exists R.A \equiv_4^A \top$).

Theorem 11. *Checking whether $\Sigma \models_4^A C(a)$, where Σ is formed out by $\mathcal{AL}$ assertions and C is an $\mathcal{ALE}_2^-$ concept, is a coNP-hard problem.* ⊣

Theorem 10 is certainly a positive result, as the $\exists R.C$ construct is very useful in the query language. This result, and in general a sequent calculus approach, suggests a detailed investigation about the computational complexity of the instance checking problem in those cases where the knowledge base language and the query language are different.

The idea of distinguishing between the knowledge base language and the query language is certainly not new in the DL area [4, 14] and has given positive results. We claim that the use of a sequent calculus framework could be better, due to its intrinsic distinction between knowledge base part and query part, than the up to now refutation based methods. As a consequence, decision procedures based on $\mid\!\sim_2$ seems to be more suitable than the (refutation based) decision procedure based on $\mid\!\sim'_2$.

A practical consideration: From an inspection on proof trees with respect to type A and type B semantics the following theorem can easily be shown[8].

Theorem 12. *Let $\Gamma \rightarrow \Delta$ be a sequent. If there is no common role symbol, concept symbol or object symbol between Γ and Δ, then $\Gamma \rightarrow \Delta$ is not provable w.r.t. type A and type B semantics.* $\dashv$

Hence, suppose that at step n of a proof we are considering sequent $\Gamma \rightarrow \Delta$. We can split Γ into two partitions Γ_1 and Γ_2, where Γ_2 is the set of assertions of Γ with no concept, role and object symbols appearing in Δ and $\Gamma_1 = \Gamma \setminus \Gamma_2$. Now, Γ_2 will be never involved in an axiom at step $n + i$ ($i \geq 0$). That is, Γ_2 is a subset of the *irrelevant* facts in Γ with respect to Δ. In fact, the following holds: $\Gamma \mid\!\sim_4^A \Delta$ iff $\Gamma_1 \cup \Gamma_2 \mid\!\sim_4^A \Delta$ iff $\Gamma_1 \mid\!\sim_4^A \Delta$. Similarly for $\mid\!\sim_4^B$. Since for huge knowledge bases, as they are in the DL-based DR model, certainly $|\Gamma_2| \gg |\Gamma_1|$ holds, a significant improvement of the deduction process can be obtained.

7 Conclusions

Four-valued DLs were introduced mainly for computational reasons, *i.e.* for designing DLs with tractable subsumption algorithms. More recently, four-valued DLs were presented in the context of multimedia document retrieval in order both to deal with inconsistent knowledge bases and to model an entailment relation with a flavour of "relevance" between premises and conclusion. Unfortunately, the known subsumption algorithms for four-valued DLs, based on "structural" subsumption, do not work with respect to type A semantics. Moreover, they are unsuitable for solving the instance checking problem, as instance checking is more difficult than subsumption.

We have presented an alternative calculus for reasoning in four-valued DLs with the aim to solve the above problems. It is based on a sequent calculus solving the instance checking problem and, thus, the subsumption problem. Moreover, we have seen that the standard constraint propagation methods for two-valued knowledge base satisfiability are a special case within our framework.

We have presented several complexity results with respect to type A and type B semantics. These show that the price we must pay for a stronger four-valued

[8] This theorem can be seen as a consequence of a four-valued version of Craig's interpolation theorem [8, 9].

semantics, *i.e.* of type A, than type B semantics is a worsening of the computational complexity. Moreover, the complexity results show that weakening classical semantics does generally not imply an improvement of the computational complexity. We have also presented a positive result where reasoning is tractable for both type A and type B semantics. It suggests us to explore those cases in which the knowledge base language differs from the query language. A sequent calculus seems to be a good candidate in such cases, as it distinguishes between assertions of the knowledge base and the query, independently of the chosen semantics (two-valued, four-valued).

Acknowledgements

This work has been carried out in the context of the project FERMI 8134 - "Formalization and Experimentation in the Retrieval of Multimedia Information", funded by the European Community under the ESPRIT Basic Research scheme.

References

1. Alan R. Anderson and Nuel D. Belnap. *Entailment - the logic of relevance and necessity*. Princeton University Press, Princeton, NJ, 1975.
2. Nuel D. Belnap. A useful four-valued logic. In Gunnar Epstein and J. Michael Dunn, editors, *Modern uses of multiple-valued logic*, pages 5–37. Reidel, Dordrecht, NL, 1977.
3. Alexander Borgida. Structural subsumption: What is it and why is it important? In *AAAI Fall Symposium:Issues in Description Logics*, pages 14–18, 1992.
4. Martin Buchheit, A. Manfred Jeusfeld, Werner Nutt, and Martin Staudt. Subsumption between queries to object-oriented databases. *Information Systems*, 19(1):33–54, 1994. Special issue on Extending Database Technology, EDBT'94.
5. Francesco M. Donini, Bernhard Hollunder, Maurizio Lenzerini, A. Marchetti Spaccamela, Daniele Nardi, and Werner Nutt. The complexity of existential quantification in concept languages. *Artificial Intelligence*, 2-3:309–327, 1992.
6. Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt. Tractable concept languages. In *Proceedings of IJCAI-91, 12th International Joint Conference on Artificial Intelligence*, pages 458–463, Sidney, Australia, 1991.
7. Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. From subsumption to instance checking. Technical Report 15.92, Università degli studi di Roma "La Sapienza". Dipartimento di informatica e sistemistica, Rome, Italy, 1992.
8. Melvin Fitting. *First-Order Logic and Automated Theorem Proving*. Springer-Verlag, 1990.
9. Jean H. Gallier. *Logic for Computer Science: Foundations of Automatic Theorem Proving*. Harper & Row Publishers, New York, 1986.
10. Michael R. Garey and David S. Johnson. *Computers and intractability. A guide to the theory of NP-completeness*. Freeman and Company, New York, NY, 1979.
11. G. Gentzen. Untersuchungen über das logische Schliessen. *Mathematische Zeitschrift*, 39:176–210,405–431, 1935.

12. C. A. Gobel, C. Haul, and S. Bechhofer. Describing and classifying multimedia using the description logic GRAIL. In *Proceedings of the SPIE Conference on Storage and Retrieval for Still Images and Video Databases IV (SPIE-95)*, pages 132–143, San Jose, CA, February 1995.
13. Bernhard Hollunder, Werner Nutt, and Manfred Schmidt-Schauß. Subsumption algorithms for concept description languages. In *Proc. of ECAI-90, 9th European Conference on Artificial Intelligence*, pages 348–353, Stockholm, Sweden, 1990.
14. Maurizio Lenzerini and Andrea Schaerf. Concept languages as query languages. In *Proc. of the 9th Nat. Conf. on Artificial Intelligence (AAAI-91)*, pages 471–476, 1991.
15. Hector J. Levesque. A logic of implicit and explicit belief. In *Proc. of the 4th Nat. Conf. on Artificial Intelligence (AAAI-84)*, pages 198–202, Austin, TX, 1984.
16. Hector J. Levesque and Ronald J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3:78–93, 1987.
17. Carlo Meghini, Fabrizio Sebastiani, Umberto Straccia, and Costantino Thanos. A model of information retrieval based on a terminological logic. In *Proceedings of SIGIR-93, 16th International Conference on Research and Development in Information Retrieval*, pages 298–307, Pittsburgh, PA, 1993.
18. Carlo Meghini and Umberto Straccia. Extending a description logic to cope with the completeness of multimedia documents. In *Proc. of the 12th European Conf. on Artificial Intelligence (ECAI-96): Workshop on Knowledge Representation for Interactive Multimedia Systems*, pages 42–50, Budapest, Hungary, 1996.
19. Carlo Meghini and Umberto Straccia. A relevance terminological logic for information retrieval. In *Proceedings of SIGIR-96, 19th International Conference on Research and Development in Information Retrieval*, pages 197–205, Zurich, Switzerland, 1996.
20. Bernhard Nebel. *Reasoning and revision in hybrid representation systems.* Springer, Heidelberg, FRG, 1990.
21. Peter F. Patel-Schneider. A four-valued semantics for frame-based description languages. In *Proceedings of AAAI-86, 5th Conference of the American Association for Artificial Intelligence*, pages 344–348, Philadelphia, PA, 1986.
22. Peter F. Patel-Schneider. A hybrid, decidable, logic-based knowledge representation system. *Computational Intelligence*, 3:64–77, 1987.
23. Peter F. Patel-Schneider. A four-valued semantics for terminological logics. *Artificial Intelligence*, 38:319–351, 1989.
24. Andrea Schaerf. Reasoning with individuals in concept languages. *Data and Knowledge Engineering*, 13(2):141–176, 1994.
25. Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48:1–26, 1991.
26. Fabrizio Sebastiani. A probabilistic terminological logic for modelling information retrieval. In *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*, pages 122–130, Dublin, IRL, 1994. Published by Springer Verlag, Heidelberg, FRG.
27. Fabrizio Sebastiani and Umberto Straccia. A computationally tractable terminological logic. In *Proceedings of SCAI-91, 3rd Scandinavian Conference on Artificial Intelligence*, pages 307–315, Roskilde, Denmark, 1991.
28. Gerd Wagner. Ex contradictione nihil sequitur. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI-91)*, pages 538–543, Sydney, Australia, 1991.