World Scientific
www.worldscientific.com

DESCRIPTION LOGICS OVER LATTICES

UMBERTO STRACCIA

Istituto di Scienza e Tecnologia dell'Informazione "A. Faedo", Via G. Moruzzi 1
Pisa, I-56124, ITALY
straccia@isti.cnr.it

It is generally accepted that knowledge based systems would be smarter if they could
manage uncertainty and/or imprecision. In this paper we extend Description Logics, well-
known logics for managing structured knowledge, allowing to express that a sentence is
not just true or false, but true to some degree, which is taken from a certainty lattice.

Keywords: Description Logics, many-valued

1. Introduction

In the last decade a substantial amount of work has been carried out in the context of
Description Logics[1] (DLs)[a]. DLs are a logical reconstruction of the so-called frame-
based knowledge representation languages, with the aim of providing a simple well-
established Tarski-style declarative semantics to capture the meaning of the most
popular features of structured representation of knowledge. A main point is that
DLs are considered as to be attractive logics in knowledge based applications as they
are a good compromise between expressive power and computational complexity.

Despite their growing popularity, relatively little work has been carried out in
extending them to the management of uncertain and imprecise information[b]. This
is a well-known and important issue whenever the real world information to be
represented is of imperfect nature.

In DLs, the problem has attracted the attention of some researchers and some
frameworks have been proposed, which differ in the underlying notion of uncertainty
and imprecision: *e.g.* probability theory[4,5,8,9,21], possibility theory[7], metric spaces[12],
many-valued[15,16,17,22] and fuzzy theory[3,6,13,19,23,24,25,26,27,29,30].

In this paper we extend DLs allowing to express that a sentence is not just true
or false like in classical DLs, but certain to some degree, which is taken from a
certainty lattice. The certainty degree dictates to what extend (how certain it is
that) a sentence is true. The adopted approach is more general than the fuzzy logic

---

[a]Description Logics have also been referred to as Terminological Logics, Concept Logics, KL-
ONE-like languages. The web page of the description logic community is found at address
http://dl.kr.org/dl.
[b]Comparing with other formalisms -notably logic programming (see, *e.g.*[10,11]).

based approach[25], as it subsumes it (just take the lattice over the real unit interval $[0, 1]$ with order $\leq$) and four-valued DLs as[17,22], but is orthogonal to almost all other approaches. A feature of the lattice approach is that it gives us the possibility to address both *quantitative* reasoning (by relying *e.g.* on $[0, 1]$ or subsets of rational numbers like $\{0, \frac{1}{n-1}, \ldots \frac{n-2}{n-1}, 1\}$, for natural number n), as well as *qualitative* uncertainty reasoning (by relying *e.g.* on {false, likelyfalse, unknown, likelytrue, true}, in increasing order). From a computational point of view, it is still possible to develop a tableau calculus in the style of almost all DLs and, under reasonable conditions, the computational complexity does not change, which is especially important as usually, reasoning under uncertainty is more involved than the classical case (see, *e.g.*[18]).

We proceed as follows. In the next section, we recall some fundamental notions about DLs. In Section 3 we describe our DL extension to manage uncertain sentences, while in Section 4, we address the computational aspect of reasoning in it. Finally, Section 6 concludes the paper.

2. A Quick Look to $\mathcal{ALC}$

The specific DL we extend is $\mathcal{ALC}$, a significant representative of DLs[1].

Consider three alphabets of symbols, *primitive concepts* (denoted A), *primitive roles* (denoted R) and *individuals* (denoted a and b)[c]. A *concept* (denoted C or D) of the language $\mathcal{ALC}$ is build out from *primitive concepts* A, the *top concept* $\top$, the *bottom concept* $\bot$ and according the following syntax rule:

$$
\begin{aligned}
C, D \longrightarrow\ & C \sqcap D| \text{ (concept conjunction)}\\
& C \sqcup D| \text{ (concept disjunction)}\\
& \neg C| \text{ (concept negation)}\\
& \forall R.C| \text{ (universal quantification)}\\
& \exists R.C \text{ (existential quantification)}.
\end{aligned}
$$

An *interpretation* $\mathcal{I}$ is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consisting of a non empty set $\Delta^{\mathcal{I}}$ (called the *domain*) and of an *interpretation function* $\cdot^{\mathcal{I}}$ mapping different individuals into different elements of $\Delta^{\mathcal{I}}$ (called *unique name assumption*), primitive concepts into subsets of $\Delta^{\mathcal{I}}$ and primitive roles into subsets of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation of complex concepts is defined as usual:

$$
\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}}\\
\bot^{\mathcal{I}} &= \emptyset\\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}}\\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}}\\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}\\
(\forall R.C)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} : \forall d'.(d, d') \notin R^{\mathcal{I}} \text{ or } d' \in C^{\mathcal{I}}\}\\
(\exists R.C)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} : \exists d'.(d, d') \in R^{\mathcal{I}} \text{ and } d' \in C^{\mathcal{I}}\}\ .
\end{aligned}
$$

Two concepts C and D are *equivalent* (denoted $C \equiv D$) when $C^{\mathcal{I}} = D^{\mathcal{I}}$ for all interpretations $\mathcal{I}$ (*e.g.* $\exists R.C \equiv \neg \forall R.\neg C$). An *assertion* (denoted α) is an expression

---

[c]Metavariables may have a subscript or a superscript.

$a{:}C$, with informal meaning "a is an instance of C", or an expression $(a, b){:}R$, with informal meaning "(a, b) is an instance of R". A *primitive assertion* is either an assertion of the form $a{:}A$, where A is a primitive concept, or an assertion of the form $(a, b){:}R$. An interpretation $\mathcal{I}$ *satisfies* $a{:}C$ (resp. $(a, b){:}R$) iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$ (resp. $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$). Let A and C be a primitive concept and a concept, respectively. A *terminological axiom* (denoted τ) is either a concept specialization or a concept definition. A *concept specialization* is an expression of the form $A{<}C$, while a *concept definition* is an expression of the form $A{:}{=}C$. An interpretation $\mathcal{I}$ *satisfies* $A{<}C$ iff $A^{\mathcal{I}} \subseteq C^{\mathcal{I}}$, while $\mathcal{I}$ *satisfies* $A{:}{=}C$ iff $A^{\mathcal{I}} = C^{\mathcal{I}}$. A finite set K of assertions and terminological axioms is a *Knowledge Base* (KB). With K_A we denote the set of assertions in K, whereas with K_T we denote the set of terminological axioms in K, also called a *terminology*. A KB K is *purely assertional* if $K_T = \emptyset$. Further, we assume that a terminology K_T is such that no concept A appears more than once on the left hand side of a terminological axiom $\tau \in K_T$ and that no cyclic definitions are present in K_T[d]. An interpretation $\mathcal{I}$ *satisfies* (*is a model of*) a KB K iff $\mathcal{I}$ satisfies each element in K. A KB K *entails* an assertion α (denoted $K \models \alpha$) iff every model of K also satisfies α. The problem of determining whether $K \models \alpha$ is called *entailment problem*, while the problem of determining whether K is satisfiable is called *satisfiability problem*. It is well known (see, *e.g.*[1]) that in $\mathcal{ALC}$:

$$K \models (a, b){:}R \text{ iff } (a, b){:}R \in K$$
$$K \models a{:}C \text{ iff } K \cup \{a{:}\neg C\} \text{ unsatisfiable .} \tag{1}$$

There exists a well known technique based on constraint propagation solving the satisfiability problem[1]. Furthermore, we can restrict our attention to purely assertional KBs, by expanding K to K' and substituting every primitive concept occurring in K, which is defined in K', with its defining term in K'. Informally, the *expansion of a KB K* is as follows[14]: replace each concept specialization $A{<}C \in K_T$ with $A{:}{=}C \sqcap A^*$ (A^* is a new primitive concept); then expand the right-hand side of every concept definition by replacing a primitive concept with its definition until there remain only undefined concepts in the second arguments of concept definitions; and finally, replace in K_A all primitive concepts with their definitions. The transformation has the nice property that $K \models \alpha$ iff $K'_A \models \alpha'$, where α' is obtained by replacing every primitive concept occurring in α, which is defined in K'_T, with its defining term in K'_T. This allows us to restrict our attention to purely assertional KBs only (but, the expansion process can be exponential[14]).

3. The logic $\mathcal{L}$-$\mathcal{ALC}$

Let $\mathcal{L} = \langle \mathcal{T}, \preceq \rangle$ be a *certainty lattice* (a complete lattice), where $\mathcal{T}$ is a set of certainty values and $\preceq$ is a partial order over $\mathcal{T}$. Let $\otimes$ and $\oplus$ be the meet and join operators induced by $\preceq$, respectively. Let f and t be the least and greatest element in $\mathcal{T}$, respectively. We also assume that there is a function from $\mathcal{T}$ to $\mathcal{T}$, called *negation function* (denoted $\neg$) that is anti-monotone w.r.t. $\preceq$ and satisfies

---

[d]We say that A *directly uses* primitive concept B in K_T, if there is $\tau \in K_T$ such that A is on the left hand side of τ and B occurs in the right hand side of τ. Let *uses* be the transitive closure of the relation directly uses in K_T. K_T is *cyclic* iff there is A such that A uses A in K_T.

$\neg\neg\alpha = \alpha, \forall \alpha \in \mathcal{T}$. The main idea is that an assertion $a{:}C$, rather being interpreted as either true or false, will be mapped into a certainty value c in $\mathcal{T}$. The intended meaning is that c indicates to which extend (how certain it is that) 'a is a C'.

Typical certainty lattices are (given a set of real values $\mathcal{T}$, consider $\mathcal{L}_{\mathcal{T}} = \langle \mathcal{T}, \leq \rangle$)

Classical 0-1: $\mathcal{L}_{\{0,1\}}$ corresponds to the classical truth-space, where 0 stands for 'false', while 1 stands for 'true';

Fuzzy: $\mathcal{L}_{[0,1]}$, which relies on the unit real interval, is quite frequently used as certainty lattice. In $\mathcal{L}_{[0,1]}$, $\neg\alpha = 1 - \alpha$ is quite typical;

Four-valued: Another frequent certainty lattice is Belnap's $\mathcal{FOUR}^2$, where $\mathcal{T}$ is $\{f, t, u, i\}$ with $f \preceq u \preceq t$ and $f \preceq i \preceq t$. Here, u stands for 'unknown', whereas i stands for inconsistency. We denote the lattice as $\mathcal{L}_B$. Additionally, besides $\neg f = t$, we have $\neg u = u$ and $\neg i = i$;

Many-valued: $\mathcal{L} = \langle \{0, \frac{1}{n-1}, \ldots \frac{n-2}{n-1}, 1\}, \leq \rangle$, n positive integer. A special case is $\mathcal{L}_4$, where $\mathcal{T}$ is $\{f, lf, lt, t\}$ with $f \preceq lf \preceq lt \preceq t$. Here, lf stands for 'likely false', whereas lt stands for 'likely true'. Besides $\neg f = t$, we have $\neg lf = lt$;

Belief-Doubt: A further popular lattice allows us to reason about *belief and doubt*. Indeed, the idea is to take any lattice $\mathcal{L}$, and to consider the cartesian product $\mathcal{L} \times \mathcal{L}$. For any pair $(b, d) \in \mathcal{L} \times \mathcal{L}$, b indicates the degree of *belief* a reasoning agent has about a sentence s, while d indicates the degree of *doubt* the agent has about s. The order on $\mathcal{L} \times \mathcal{L}$ is determined by $(b, d) \preceq (b', d')$ iff $b \preceq b'$ and $d' \preceq d$, *i.e.* belief goes up, while doubt goes down. The minimal element is (f, t) (no belief, maximal doubt), while the maximal element is (t, f) (maximal belief, no doubt). Negation is given by $\neg(b, d) = (d, b)$ (exchange belief with doubt). We indicate this lattice with $\bar{\mathcal{L}}$.

For a certainty lattice $\mathcal{L} = \langle \mathcal{T}, \preceq \rangle$, an $\mathcal{L}$-*interpretation* is now a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is, as for the classical case, the *domain*, whereas $\cdot^{\mathcal{I}}$ is an *interpretation function* mapping

- individuals as for the classical case, *i.e.* $a^{\mathcal{I}} \neq b^{\mathcal{I}}$, if $a \neq b$;
- a concept C into a function $C^{\mathcal{I}} : \Delta^{\mathcal{I}} \to \mathcal{T}$;
- a role R into a function $R^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \to \mathcal{T}$.

For ease with interpretation we mean always an $\mathcal{L}$-interpretation, for some certainty lattice $\mathcal{L}$. As anticipated above, if $d \in \Delta^{\mathcal{I}}$ is an object of the domain $\Delta^{\mathcal{I}}$ then $C^{\mathcal{I}}(d)$ gives us the degree of certainty of being the object d an instance of the concept C under the interpretation $\mathcal{I}$. Similarly for roles. The interpretation

function $\cdot^{\mathcal{I}}$ has to satisfy the following equations: for all $d \in \Delta^{\mathcal{I}}$,

$$\begin{aligned}
\top^{\mathcal{I}}(d) &= t \\
\bot^{\mathcal{I}}(d) &= f \\
(C \sqcap D)^{\mathcal{I}}(d) &= C^{\mathcal{I}}(d) \otimes D^{\mathcal{I}}(d) \\
(C \sqcup D)^{\mathcal{I}}(d) &= C^{\mathcal{I}}(d) \oplus D^{\mathcal{I}}(d) \\
(\neg C)^{\mathcal{I}}(d) &= \neg C^{\mathcal{I}}(d) \\
(\forall R.C)^{\mathcal{I}}(d) &= \bigotimes_{d' \in \Delta^{\mathcal{I}}} \{\neg R^{\mathcal{I}}(d, d') \oplus C^{\mathcal{I}}(d')\} \\
(\exists R.C)^{\mathcal{I}}(d) &= \bigoplus_{d' \in \Delta^{\mathcal{I}}} \{R^{\mathcal{I}}(d, d') \otimes C^{\mathcal{I}}(d')\} \ .
\end{aligned}$$

Note that the semantics of $\exists R.C$ is the result of viewing $\exists R.C$ as the open first order formula $\exists y.R(x, y) \wedge \bar{C}(y)$ (where $\bar{C}$ is the translation of C into first-order logic) and $\exists$ is viewed as a disjunction over the elements of the domain. Similarly, the semantics of $\forall R.C$ is related to $\forall y.\neg R(x, y) \vee \bar{C}(y)$, where $\forall$ is viewed as a conjunction over the elements of the domain. The definition of concept *equivalence* is like for $\mathcal{ALC}$. As for classical $\mathcal{ALC}$, dual relationships between concepts hold: *e.g.* $\top \equiv \neg \bot$, $(C \sqcap D) \equiv \neg(\neg C \sqcup \neg D)$ and $(\forall R.C) \equiv \neg(\exists R.\neg C)$.

An $\mathcal{L}$-*assertion* (denoted ψ) is an expression $\langle \alpha \succeq c \rangle$ or $\langle \alpha \preceq c' \rangle$, where α is an $\mathcal{ALC}$ assertion and $c, c' \in \mathcal{T}$. From a semantics point of view, an $\mathcal{L}$-assertion $\langle \alpha \preceq c \rangle$ constrains the certainty-value of α to be less or equal to c (similarly for $\succeq$). An interpretation $\mathcal{I}$ *satisfies* $\langle a{:}C \succeq c \rangle$ (resp. $\langle (a, b){:}R \succeq c \rangle$) iff $C^{\mathcal{I}}(a^{\mathcal{I}}) \succeq c$ (resp. $R^{\mathcal{I}}(a^{\mathcal{I}}, b^{\mathcal{I}}) \succeq c$). Similarly for $\preceq$. Two $\mathcal{L}$-assertions ψ_1 and ψ_2 are *equivalent* (denoted $\psi_1 \equiv \psi_2$) iff they are satisfied by the same set of interpretations. Notice that $\langle a{:}\neg C \succeq c \rangle \equiv \langle a{:}C \preceq \neg c \rangle$. A *primitive* $\mathcal{L}$-*assertion* is a $\mathcal{L}$-assertion involving a primitive assertion only. One might wonder why we do not allow expressions of the form $\langle \alpha \succ c \rangle$ or the form $\langle \alpha \prec c \rangle$. The reason relies on the observation that it is quite hard to imagine situations in which we are able to assert such *strict* $\succ, \prec$ relations. However, from a technical point of view it is easy to manage these constraints as well. For ease of presentation, we leave them out.

Concerning terminological axioms, an interpretation $\mathcal{I}$ *satisfies* $A \!<\! C$ iff $\forall d \in \Delta^{\mathcal{I}}, A^{\mathcal{I}}(d) \preceq C^{\mathcal{I}}(d)$, while $\mathcal{I}$ *satisfies* $A := C$ iff $\forall d \in \Delta^{\mathcal{I}}, A^{\mathcal{I}}(d) = C^{\mathcal{I}}(d)$. In $\mathcal{L}$-$\mathcal{ALC}$, a *knowledge base* is a finite set of $\mathcal{L}$-assertions and terminological axioms. With Σ_A we denote the set of $\mathcal{L}$-assertions in Σ, with Σ_T we denote the set of terminological axioms in Σ (the terminology), if $\Sigma_T = \emptyset$ then Σ is *purely assertional*, and we assume that a terminology Σ_T is such that no concept A appears more than once on the left hand side of a terminological axiom in Σ_T and that no cyclic definitions are present in Σ_T. An interpretation $\mathcal{I}$ *satisfies* (is *a model of*) a knowledge base Σ iff $\mathcal{I}$ satisfies each element of Σ. A KB Σ $\mathcal{L}$-*entails* an $\mathcal{L}$-assertion ψ (denoted $\Sigma \models_{\mathcal{L}} \psi$) iff every model of Σ also satisfies ψ. Finally, given a KB Σ and an assertion α, it is of interest to compute α's best lower and upper certainty-value bounds. To this aim, the *greatest lower bound* of α w.r.t. Σ (denoted $glb(\Sigma, \alpha)$) is $\bigoplus \{c : \Sigma \models_{\mathcal{L}} \langle \alpha \succeq c \rangle \}$, while the *least upper bound* of α with respect to Σ (denoted $lub(\Sigma, \alpha)$) is $\bigotimes \{c : \Sigma \models_{\mathcal{L}} \langle \alpha \preceq c \rangle\}$ ($\bigoplus \emptyset = f, \bigotimes \emptyset = t$). Determining the *lub* and the *glb* is called the *Best Certainty-Value Bound* (BCVB) problem. Note that from

$\Sigma \models_{\mathcal{L}} \langle a{:}C \preceq c \rangle$ iff $\Sigma \models_{\mathcal{L}} \langle a{:}\neg C \succeq \neg c \rangle$, $lub(\Sigma, a{:}C) = \neg glb(\Sigma, a{:}\neg C)$ can be shown. The same reduction to glb does not hold for $lub(\Sigma, (a, b){:}R)$ as $(a, b){:}\neg R$ is not an expression of our language[e].

$\mathcal{L}\text{-}\mathcal{ALC}$ is a sound extension of $\mathcal{ALC}$. In fact, assume that in Σ no $\langle (a, b){:}R \preceq c \rangle$ occurs. We leave these $\mathcal{L}$-assertions out, as role negation is not present in $\mathcal{ALC}$. Consider the following transformation $\sharp(\cdot)$ from $\mathcal{L}$-assertions to assertions: $\sharp \langle \alpha \succeq c \rangle \mapsto \alpha$, $\sharp \langle a{:}C \preceq c \rangle \mapsto a{:}\neg C$, and $\sharp \Sigma = \{\sharp \psi : \psi \in \Sigma\} \cup \Sigma_T$. It can be shown that

Proposition 1. *Let Σ be a KB in which no $\langle (a, b){:}R \preceq c \rangle$ occurs and consider $\langle \alpha \succeq c \rangle$. If $\Sigma \models_{\mathcal{L}} \langle \alpha \succeq c \rangle$ then $\sharp \Sigma \models \alpha$.*

The converse does not hold in general and depends on $\mathcal{L}$, e.g. in $\mathcal{L}_{[0,1]}\text{-}\mathcal{ALC}$, for all $c \succ f$ $\{\langle a{:}A \succeq 0.1 \rangle, \langle a{:}\neg A \sqcup B \succeq 1 \rangle\} \not\models_{\mathcal{L}} \langle a{:}A \succeq c \rangle$, whereas $\{a{:}A, a{:}\neg A \sqcup B\} \models a{:}A$. A simple 'converse' is the following. Let K be an $\mathcal{ALC}$ KB: we define $\tilde{K} = \{\langle \alpha \succeq 1 \rangle : \alpha \in K\} \cup K_T$. Then,

Proposition 2. *If $K \models \alpha$ then $\tilde{K} \models_{\mathcal{L}} \langle \alpha \succeq c \rangle$ for some $c \in \mathcal{L}$.*

4. Decision algorithms in $\mathcal{L}\text{-}\mathcal{ALC}$

Deciding satisfiability of a KB requires a calculus. Without loss of generality we consider purely assertional KBs only. We first develop a calculus in the style of the constraint propagation method, as this method is usually proposed in the context of DLs[1]. We then present a transformation of $\mathcal{L}\text{-}\mathcal{ALC}$ under linear ordered lattices into $\mathcal{ALC}$ and, thus, classical DLs reasoners may be applied in this special case.

4.1. *Tableau-like calculus*

Essentially, we generalize the calculus presented in[25] for $\mathcal{L}_{[0,1]}$, to any certainty lattice $\mathcal{L}$. We first address the entailment problem and then the BCVB problem.

To guarantee soundness and completeness of the calculus, we make the following restrictions. We assume that in the certainty lattice $\mathcal{L} = \langle \mathcal{T}, \preceq \rangle$ the set of certainty values $\mathcal{T}$ is *finite*. From a practical point of view this is a limitation we can live with, especially taking into account that computers have finite resources, and thus, only a finite set of certainty values can be represented. In particular, this includes also the case of the the rational numbers in $[0, 1] \cap \mathbb{Q}$ under a given fixed precision p a computer can work with. We also point out an error in our early version[28], in which we do not rely on this assumption. Without this assumption, the results in[28] do not hold in general.

An $\mathcal{L}$-*constraint* (denoted ψ) is inductively defined as follows: (*i*) an $\mathcal{L}$-assertion is an $\mathcal{L}$-constraint; (*ii*) if ψ and ψ' are $\mathcal{L}$-constraints, then so are $\neg \psi, \psi \wedge \psi'$ and $\psi \vee \psi'$ (*e.g.* $\langle \alpha_1 \preceq c_1 \rangle \vee \neg \langle \alpha_2 \succeq c_2 \rangle$). A *literal* is a primitive $\mathcal{L}$-assertion or its boolean negation. Without loss of generality, we assume that $\mathcal{L}$-constraints are always in

---

[e]Of course, $lub(\Sigma, (a, b){:}R) = \neg glb(\Sigma, (a, b){:}\neg R)$ holds, where $(\neg R)^{\mathcal{I}}(d, d') = \neg R^{\mathcal{I}}(d, d')$.

Table 1. Conjugated pairs.

	$\langle \alpha \not\succeq c' \rangle$	$\langle \alpha \preceq c' \rangle$
$\langle \alpha \succeq c \rangle$	$\neg(\exists c''.c'' \succeq c \wedge c'' \not\succeq c')$	$c \not\preceq c'$
$\langle \alpha \not\preceq c \rangle$	$\neg(\exists c''.c'' \not\preceq c \wedge c'' \not\succeq c')$	$\neg(\exists c''.c'' \not\preceq c \wedge c'' \preceq c')$

Negation Normal Form (NNF), where a boolean negation appears in the head of an $\mathcal{L}$-assertion only. The definition of satisfiability (of a set) of $\mathcal{L}$-constraints is easy: *e.g.* I satisfies $\neg\psi$ iff I does not satisfy ψ, while I satisfies $\psi \vee \psi'$ iff I does satisfy either ψ or ψ'. It follows that (compare to Eq. 1)

$$\begin{aligned} \Sigma \models_{\mathcal{L}} \langle \alpha \succeq c \rangle \text{ iff } \Sigma \cup \{\neg\langle \alpha \succeq c \rangle\} \text{ not satisfiable} \\ \Sigma \models_{\mathcal{L}} \langle \alpha \preceq c \rangle \text{ iff } \Sigma \cup \{\neg\langle \alpha \preceq c \rangle\} \text{ not satisfiable .} \end{aligned} \qquad (2)$$

Note that in case $\mathcal{L}$ is a total order, then we have the equivalences between $\neg\langle \alpha \succeq c \rangle$ and $\langle \alpha \prec c \rangle$, and between $\neg\langle \alpha \preceq c \rangle$ and $\langle \alpha \succ c \rangle$. For instance, this property is used in the calculus developed for fuzzy $\mathcal{ALC}$[25]. In general, these equivalences do not hold (*e.g.*, in $\mathcal{L}_B$, $\neg\langle \alpha \succeq u \rangle$ is equivalent to $\langle \alpha \preceq i \rangle$ and not to $\langle \alpha \prec u \rangle$). For ease, sometimes we write $\langle \alpha \not\succeq c \rangle$ in place of $\neg\langle \alpha \succeq c \rangle$ and $\langle \alpha \not\preceq c \rangle$ in place of $\neg\langle \alpha \preceq c \rangle$.

Our calculus, determining whether a finite set S of $\mathcal{L}$-constraints is satisfiable or not, is based on a set of constraint propagation rules transforming a set S into "simpler" satisfiability preserving sets S_i until either all S_i contain an inconsistency, called *clash* (indicating that from all the S_i no model of S can be build), or some S_i is completed and clash-free, that is, no rule can further be applied to S_i and S_i contains no clash (indicating that from S_i a model of S can be built). A set of $\mathcal{L}$-constraints S contains a *clash* (inconsistency) iff it contains a set of literals $\{\langle \alpha_j \ r_j \ c_j \rangle\}_{j \in J}$ (with $r_j \in \{\succeq, \preceq, \not\succeq, \not\preceq\}$) such that the set of constraints $\{\langle x_j \ r_j \ c_j \rangle\}_{j \in J}$ has no solution for the variables x_j in the lattice $\mathcal{L}$[f]. Of course, if $\alpha_i = \alpha_j$ then $x_i = x_j$ and if if α is either $a{:}\bot$ or $a{:}\top$ then x is replaced with f and t, respectively. For instance, S contains a clash if it contains either $\langle a{:}\bot \succeq c \rangle$ (where $c \succ f$), or $\langle a{:}\top \preceq c \rangle$ (where $c \prec t$), or $\langle a{:}\bot \not\preceq c \rangle$, or $\langle a{:}\top \not\succeq c \rangle$, or $\langle \alpha \not\succeq f \rangle$, or $\langle \alpha \not\preceq t \rangle$, or S contains a conjugated pair of $\mathcal{L}$-constraints. Each entry in Table 1 says us under which condition the row-column pair of $\mathcal{L}$-constraints is a *conjugated pair*.

For instance, in $\mathcal{L}_B$, $\langle a{:}A \succeq i \rangle$ and $\langle a{:}A \preceq u \rangle$ is a conjugated pair as $i \not\preceq u$. While in total ordered lattices checking inconsistency is easy, this may not be the case for arbitrary lattices. Given an $\mathcal{L}$-constraint ψ, with ψ^c we indicate a conjugate of ψ (if there exists one). A conjugate of an $\mathcal{L}$-constraint may be not unique, as there could be infinitely many. For instance, in $\mathcal{L}_{[0,1]}$-$\mathcal{ALC}$, any $\langle a{:}A \preceq c \rangle$ with $c \prec c'$ is a conjugate of $\langle a{:}A \succeq c' \rangle$.

---

[f]A solution for x_j is a certainty value c'_j such that all constraints in $\{\langle c'_j \ r_j \ c_j \rangle\}_{j \in J}$ are satisfied, *i.e.* for all $j \in J$, $c'_j \ r_j \ c_j$ holds in $\mathcal{L}$.

There are obvious tableau rules for the boolean connectives $\wedge$ and $\vee$.

For each connective $\sqcap, \sqcup, \neg, \forall, \exists$ there is a rule for each relation $\succeq, \preceq, \not\succeq$ and $\not\preceq$.

In what follows, for any $c \in \mathcal{T}$, with $D_{\mathcal{L}}(c)$ we indicate the set $D_{\mathcal{L}}(c) = \min\{(c_1, c_2) \in \mathcal{T} \times \mathcal{T} : c_1 \oplus c_2 \succeq c\}$ where the order over pairs is $(c_1, c_2) \preceq (c_3, c_4)$ iff $c_1 \preceq c_3$ and $c_2 \preceq c_4$. The purpose of $D_{\mathcal{L}}(c)$ is to identify meaningful candidates c_1 and c_2, making $e.g.$ a disjunction of the form $\langle a{:}A \sqcup B \succeq c \rangle$ true, whenever we reduce $\langle a{:}A \sqcup B \succeq c \rangle$ to $\{\langle a{:}A \succeq c_1 \rangle \wedge \langle a{:}B \succeq c_2 \rangle\}$. The choice is non-deterministic and there could $|D_{\mathcal{L}}(c)|$ many choices, and $|D_{\mathcal{L}}(c)|$ depends on c and $\mathcal{L}$.

Note that $(c_1, c_2) \in D_{\mathcal{L}}(c)$ iff $(c_2, c_1) \in D_{\mathcal{L}}(c)$ and $|D_{\mathcal{L}}(c)| \succeq 2$ as $\{(c, f), (f, c)\} \subseteq D_{\mathcal{L}}(c)$, $e.g.$ in $\mathcal{L}_{\{0,1\}}$, $D_{\mathcal{L}_{\{0,1\}}}(1) = \{(1, 0), (0, 1)\}$ indicating that $e.g.$ $\langle a{:}A \sqcup B \succeq c \rangle$ can be branched into either $\{\langle a{:}A \succeq 1 \rangle\}$ or $\{\langle a{:}B \succeq 1 \rangle\}$. Note also that in $\mathcal{L}_B$, we have $D_{\mathcal{L}_B}(t) = \{(t, f), (f, t), (u, i), (i, u)\}$ and, thus, $e.g.$ $\langle a{:}A \sqcup B \succeq c \rangle$ can be branched into four alternatives. If $\mathcal{L}$ has finite $\mathcal{T}$, then for any $c \in \mathcal{T}$, $D_{\mathcal{L}}(c)$ is finite as well. If $\mathcal{L}$ is a total order, like $\mathcal{L}_4$ or $\mathcal{L}_{[0,1]}$, then $D_{\mathcal{L}}(c) = \{(c, f), (f, c)\}$, for any $c \in \mathcal{T}$. $|D_{\mathcal{L}}(c)|$ may also be infinite. For instance, consider the following lattice $\mathcal{L}' = \langle \mathcal{T}, \preceq \rangle$, where $\mathcal{T} = \{f, t\} \cup \{x_i : i = 0, 1, 2, \ldots\}$ and $f \preceq x_i \preceq t$ and $x_i \not\preceq x_j$ for all $i \neq j$. Then $D_{\mathcal{L}'}(t) = \{(t, f), (f, t)\} \cup \{(x_i, x_j) : i \neq j\}$ and $|D_{\mathcal{L}'}(t)|$ is infinite.

We call a certainty lattice $\mathcal{L}$ *safe* iff (i) for any $c \in \mathcal{T}$, $D_{\mathcal{L}}(c)$ is finite; and (ii) the decision problem whether a set of constraints is inconsistent is decidable.

Our decision procedure is guaranteed to terminate, whenever we restrict lattices to be safe. This is not a severe limitation as it is hard to imagine an application involving unsafe lattices like $\mathcal{L}'$ above. Note that, more generally, it is easily verified that a lattice $\mathcal{L}$ having a finite set of incomparable certainty values is save, where two elements $c, c' \in \mathcal{T}$ are *incomparable* iff neither $c \preceq c'$ nor $c' \preceq c$.

We present rules for $\sqcap, \sqcup, \neg, \forall, \exists$, and $\succeq$ and $\not\succeq$ only. The rules for $\preceq, \not\preceq$ can be derived from $\succeq$ and $\not\succeq$, respectively. Indeed, the rules for $\langle a{:}C \preceq c \rangle$, $\langle a{:}C \sqcup D \preceq c \rangle$, $\langle a{:}C \sqcap D \preceq c \rangle$, $\langle a{:}\exists R.C \preceq c \rangle$ and $\langle a{:}\forall R.C \preceq c \rangle$ can be derived from the equivalent expressions $\langle a{:}\neg C \succeq \neg c \rangle$, $\langle a{:}\neg C \sqcap \neg D \succeq \neg c \rangle$, $\langle a{:}\neg C \sqcup \neg D \succeq \neg c \rangle$, $\langle a{:}\forall R.\neg C \succeq \neg c \rangle$ and $\langle a{:}\exists R.\neg C \succeq \neg c \rangle$, respectively. Similarly for $\not\preceq$ ($e.g.$ we have the following equivalence: $\langle a{:}C \not\preceq c \rangle \equiv \neg\langle a{:}C \preceq c \rangle \equiv \neg\langle a{:}\neg C \succeq \neg c \rangle \equiv \langle a{:}\neg C \not\succeq \neg c \rangle$).

The rules below rely on the following properties over a lattice $\mathcal{L} = \langle \mathcal{T}, \preceq \rangle$: for any $c, c_1, c_2 \in \mathcal{T}$, (i) $c_1 \otimes c_2 \succeq c$ iff $c_1 \succeq c$ and $c_2 \succeq c$; (ii) $c_1 \otimes c_2 \not\succeq c$ iff $c_1 \not\succeq c$ or $c_2 \not\succeq c$; (iii) $c_1 \oplus c_2 \succeq c$ iff $c_1 \succeq c'$ and $c_2 \succeq c''$, for $(c', c'') \in D_{\mathcal{L}}(c)$; and (iv) $c_1 \oplus c_2 \not\succeq c$ iff not $(c_1 \oplus c_2 \succeq c)$ iff for all $(c', c'') \in D_{\mathcal{L}}(c)$, either $c_1 \not\succeq c'$ or $c_2 \not\succeq c''$.

In the rules we assume that a new constraint is added to a constraint set S if it is not subsumed in S (a constraint $\langle \alpha \succeq c \rangle$ is *subsumed* by a constraint $\langle \alpha \succeq c' \rangle$ iff $c' \succeq c$ -similarly for the other relations $\not\succeq, \preceq$, and $\not\preceq$). We also avoid adding constraints of the form $\langle \alpha \succeq f \rangle$ and $\langle \alpha \preceq t \rangle$.

(1) $S \rightarrow_{\sqcap_{\succeq}} S \cup \{\langle a{:}C \succeq c \rangle \wedge \langle a{:}D \succeq c \rangle\}$, if $\psi = \langle a{:}C \sqcap D \succeq c \rangle \in S$ and the $(\sqcap_{\succeq})$ rule has not yet been applied to $\psi \in S$

(2) $S \rightarrow_{\sqcap_{\not\succeq}} S \cup \{\langle a{:}C \not\succeq c \rangle \vee \langle a{:}D \not\succeq c \rangle\}$, if $\psi = \langle a{:}C \sqcap D \not\succeq c \rangle \in S$ and the $(\sqcap_{\not\succeq})$ rule has not yet been applied to $\psi \in S$

(3) $S \to_{\sqcup_{\succeq}} S \cup \{\psi'\}$, if $\psi = \langle a{:}C_1 \sqcup C_2 \succeq c\rangle \in S$, the $(\sqcup_{\succeq})$ rule has not yet been applied to $\psi \in S$ and $\psi' = \bigvee_{(c_1,c_2)\in D_{\mathcal{L}}(c)} \langle a{:}C_1 \succeq c_1\rangle \wedge \langle a{:}C_2 \succeq c_2\rangle$

(4) $S \to_{\sqcup_{\not\succeq}} S \cup \{\psi'\}$, if $\psi = \langle a{:}C_1 \sqcup C_2 \not\succeq c\rangle \in S$, the $(\sqcup_{\not\succeq})$ rule has not yet been applied to $\psi \in S$ and $\psi' = \bigwedge_{(c_1,c_2)\in D_{\mathcal{L}}(c)}(\langle a{:}C_1 \not\succeq c_1\rangle \vee \langle a{:}C_2 \not\succeq c_2\rangle)$

(5) $S \to_{\forall_{\succeq}} S \cup \{\psi'\}$, if $\{\langle a{:}\forall R.C \succeq c\rangle, \psi^c\} \subseteq S$, $\psi = \langle (a,b){:}R \preceq \neg c\rangle$, the $(\forall_{\succeq})$ rule has not yet been applied in S to the pair $(\langle a{:}\forall R.C \succeq c\rangle, \psi)$ and $\psi' = \bigvee_{(c_1,c_2)\in D_{\mathcal{L}}(c)\setminus\{(c,f)\}} \langle (a,b){:}R \preceq \neg c_1\rangle \wedge \langle b{:}C \succeq c_2\rangle$

(6) $S \to_{\forall_{\not\succeq}} S \cup \{\psi'\}$, if $\psi = \langle a{:}\forall R.C \not\succeq c\rangle, \in S$, the $(\forall_{\not\succeq})$ rule has not yet been applied to $\psi \in S$ and $\psi' = \bigwedge_{(c_1,c_2)\in D_{\mathcal{L}}(c)} \langle (a,b){:}R \npreceq \neg c_1\rangle \vee \langle b{:}C \not\succeq c_2\rangle$, for a new constant b

(7) $S \to_{\exists_{\succeq}} S \cup \{\langle (a,b){:}R \succeq c\rangle \wedge \langle b{:}C \succeq c\rangle\}$, if $\psi = \langle a{:}\exists R.C \succeq c\rangle \in S$, b new constant and the $(\exists_{\succeq})$ rule has not yet been applied to $\psi \in S$

(8) $S \to_{\exists_{\not\succeq}} S \cup \{\psi'\}$, if $\{\langle a{:}\exists R.C \succeq c\rangle, \psi^c\} \subseteq S$, $\psi = \langle (a,b){:}R \not\succeq c\rangle$, the $(\exists_{\not\succeq})$ rule has not yet been applied in S to the pair $(\langle a{:}\exists R.C \succeq c\rangle, \psi)$ and $\psi' = \langle b{:}C \not\succeq c_2\rangle)$.

Some of the above rules deserve some explanation. The $(\sqcup_{\succeq})$ rule is a generalization of the classical disjunction rule. The main difference is that, depending on the lattice $\mathcal{L}$, there may be more than the usual two branches (likely as many as $|D_{\mathcal{L}}(c)|$). For instance, in $\mathcal{L}_B$, a constraint $\langle a{:}C_1 \sqcup C_2 \succeq t\rangle \in S$ may give rise to four branches, each containing $\langle a{:}C_1 \succeq t\rangle$, $\langle a{:}C_2 \succeq t\rangle$, $\langle a{:}C_1 \succeq i\rangle \wedge \langle a{:}C_2 \succeq u\rangle$ and $\langle a{:}C_1 \succeq u\rangle \wedge \langle w{:}C_2 \succeq i\rangle$, respectively. Note that if $\mathcal{L}$ is a total order then there are at most two branches. The $(\sqcup_{\not\succeq})$ rule is derived from the $(\sqcup_{\succeq})$ rule. The $\forall_{\succeq}$ is a specialization of the $(\sqcup_{\succeq})$ rule as well and is a generalization of the classical rule for $\forall$. Indeed, according to the semantics of $\mathcal{L}\text{-}\mathcal{ALC}$, $\langle a{:}\forall R.C \succeq c\rangle$ may be viewed as a disjunction and with decomposition rule

- $S \to_{\forall'_{\succeq}} S \cup \{\langle (a,b){:}R \preceq \neg c_1\rangle, \langle b{:}C \succeq c_2\rangle\}$
  if $\langle a{:}\forall R.C \succeq c\rangle \in S$, b occurs in S and $(c_1,c_2) \in D_{\mathcal{L}}(c)$

But, observing that $(c,f) \in D_{\mathcal{L}}(c)$, we obtain a constraint set S containing $\psi = \langle (a,b){:}R \preceq \neg c\rangle$ (e.g., in the classical $\mathcal{L}_{\{t,f\}}$, with $c = t$, we have $\langle (a,b){:}R \preceq f\rangle \in S$). This constraint can only be clashed if S contains a conjugate to ψ (e.g., in the classical case $\langle (a,b){:}R \succeq t\rangle$ must be in S). This motivates the $\forall_{\succeq}$ rule as a refinement of the above $\forall'_{\succeq}$ rule. Note that the $(\forall_{\not\succeq})$ rule can be worked out by a similar argumentation, by relying on the $(\sqcup_{\not\succeq})$ rule, and is left to the reader.

A constraint set S is *complete* if no rule is applicable to it. A complete set S_2 obtained from a set S_1 by applying the above rules is called a *completion* of S_1. Note that more than one completion can be obtained. It can be verified that for save certainty lattices, the above calculus has the *termination property*, *i.e.* any completion of a finite set of $\mathcal{L}$-constraints S can be obtained after a finite number of rule applications.

Example 1. Consider $\mathcal{L}_B$, $\Sigma = \{\langle a{:}\exists R.C \succeq t\rangle, \langle a{:}\forall R.D \succeq t\rangle\}$ and $\psi = \langle a{:}\exists R.(C \sqcap D) \succeq t\rangle$. Let's show that $\Sigma \models_{\mathcal{L}} \psi$ by proving that $S = \Sigma \cup$

$\{\neg\langle a{:}\exists R.(C \sqcap D) \succeq t\rangle\}$ is unsatisfiable. We have the following sequences.

(1)	$\langle a{:}\exists R.C \succeq t\rangle$	Hypothesis:S
(2)	$\langle a{:}\forall R.D \succeq t\rangle$	
(3)	$\langle a{:}\exists R.(C \sqcap D) \not\succeq t\rangle$	
(4)	$\langle (a,b){:}R \succeq t\rangle, \langle b{:}C \succeq t\rangle$	$(\exists_{\succeq}){:}(1)$
(5)	$\langle b{:}D \succeq t\rangle\vee$	$(\forall_{\succeq}){:}(2),(4)$
	$(\langle (a,b){:}R \preceq u\rangle \wedge \langle b{:}D \succeq i\rangle)\vee$	
	$(\langle (a,b){:}R \preceq i\rangle \wedge \langle b{:}D \succeq u\rangle)$	
(6)	$\langle b{:}C \sqcap D \not\succeq t\rangle$	$(\exists_{\not\succeq}){:}(3),(4)$
(7)	$\langle b{:}C \not\succeq t\rangle \vee \langle b{:}D \not\succeq t\rangle$	$(\sqcap_{\not\succeq}){:}(6)$

(8) $\langle b{:}C \not\succeq t\rangle$ $(\vee){:}(7)$	(10) $\langle b{:}D \not\succeq t\rangle$ $(\vee){:}(7)$
(9) clash $(4),(8)$	$\Omega_1 \mid \Omega_2 \mid \Omega_3$

where the sequences Ω_i are defined as follows: for Ω_1 we have
(11) $\langle b{:}D \succeq t\rangle$ $(\vee){:}(5)$
(12) clash $(10),(11)$
for Ω_2 we have
(13) $\langle (a,b){:}R \preceq u\rangle \wedge \langle b{:}D \succeq i\rangle$ $(\vee){:}(5)$
(14) $\langle (a,b){:}R \preceq u\rangle, \langle b{:}D \succeq i\rangle$ $(\wedge){:}(13)$
(15) clash $(4),(14)$
while for Ω_3 we have
(16) $\langle (a,b){:}R \preceq i\rangle \wedge \langle b{:}D \succeq u\rangle$ $(\vee){:}(5)$
(17) $\langle (a,b){:}R \preceq i\rangle, \langle b{:}D \succeq u\rangle$ $(\wedge){:}(13)$
(18) clash $(4),(17)$

Soundness and completeness of the calculus can be shown.

Proposition 3. *For a safe certainty lattice $\mathcal{L}$, a finite set of $\mathcal{L}$-constraints S is satisfiable iff there exists a clash free completion of S.*

Proof. [Sketch] Given the termination property, it is not difficult to show by case analysis on rule applications, that the above rules are sound, *i.e.* if S is satisfiable then there is a satisfiable completion S' of S and, thus, S' contains no clash. For instance, let us show that the $(\sqcup_{\succeq})$ rule is sound. Assume that I satisfies $\langle a{:}C_1 \sqcup C_2 \succeq c\rangle \in S$. Let us show that there is $(c_1,c_2) \in D_{\mathcal{L}}(c)$ such that I satisfies both $\langle a{:}C_1 \succeq c_1\rangle$ and $\langle a{:}C_2 \succeq c_2\rangle$. By assumption, $(C_1 \sqcup C_2)^{I}(a^{I}) \succeq c$, *i.e.* $C_1^{I}(a^{I}) \oplus C_2^{I}(a^{I}) \succeq c$. Therefore, $(C_1^{I}(a^{I}), C_2^{I}(a^{I})) \in \{(c_1,c_2){:}c_1 \oplus c_2 \succeq c\}$. As a consequence, there is $(c_1,c_2) \in D_{\mathcal{L}}(c)$ such that $c_1 \preceq C_1^{I}(a^{I})$, $c_2 \preceq C_2^{I}(w^{I})$, *i.e.* I satisfies both $\langle a{:}C_1 \succeq c_1\rangle$ and $\langle a{:}C_2 \succeq c_2\rangle$.

Similarly, for $\langle a{:}\exists R.C \succeq c\rangle \in S$ we use the fact that for finite lattices, if *e.g.*

$$(\exists R.C)^{I}(d) = \bigoplus_{d' \in \Delta^{I}} \{R^{I}(d,d') \otimes C^{I}(d')\} \succeq c$$

then there is $d' \in \Delta^{I}$ such that $R^{I}(d,d') \succeq c$ and $C^{I}(d') \succeq c$, which motivates rule 7. The case for $\forall R.C$ is similar.

Vice-versa, completeness, *i.e.* if there is a completion S' of S containing no clash then S is satisfiable, can be shown by building an interpretation I from S' satisfying

S. As $S \subseteq S'$, $\mathcal{I}$ satisfies S. Roughly, given a clash-free completion S' of S, for any primitive assertion α, we collect its lower and upper bound restrictions in S': $N_1[\alpha] = \{c : \langle \alpha \succeq c \rangle \in S'\}$, and $N_2[\alpha] = \{c : \langle \alpha \not\succeq c \rangle \in S'\}$. $N_3[\alpha] = \{c : \langle \alpha \preceq c \rangle \in S'\}$, and $N_4[\alpha] = \{c : \langle \alpha \not\preceq c \rangle \in S'\}$. Since S' is clash-free, for any primitive assertion α, using $N_i[\alpha]$, we can find an appropriate $c[\alpha] \in \mathcal{T}$ such that the interpretation $\mathcal{I}$, (*i*) with domain $\Delta^{\mathcal{I}}$ being the set of individuals appearing in S', (*ii*) $a^{\mathcal{I}} = a$ for all $a \in \Delta^{\mathcal{I}}$ and (*iii*) $A^{\mathcal{I}}(a^{\mathcal{I}}) = c[a:A]$, $R^{\mathcal{I}}(a^{\mathcal{I}}, b^{\mathcal{I}}) = c[(a,b):R]$, satisfies both S' and S. For instance, suppose $\{\langle a{:}A \not\succeq c \rangle, \langle a{:}A \not\preceq c' \rangle\} \subseteq S'$ and there is no other constraint in S' involving $a{:}A$. As S' is clash-free, these two constraints are consistent, *i.e.* there is a solution for x to the set of constraints $\{\langle x \not\succeq c \rangle, \langle x \not\preceq c' \rangle\}$ and, thus, there is c'' such that $c'' \not\succeq c$ and $c'' \not\preceq c'$. Hence, in this case it suffices to chose $c[w{:}A] = c''$. $\square$

From a computational complexity point of view, under certain circumstances, which depend on the particular certainty lattice $\mathcal{L}$, the satisfiability problem is in the same complexity class (PSPACE-complete[20]) as for $\mathcal{ALC}$. Indeed, let us indicate with $|\mathcal{L}|$ the dimension of representing $\mathcal{L}$ and with $|S|$ the dimension of a constraint set S. With *combined complexity* we intend the complexity w.r.t. $|\mathcal{L}| + |S|$. We have to ensure that (*i*) $\mathcal{L}$ is safe; (*ii*) for any set of $\mathcal{L}$-constraints S and certainty value c, $|D_{\mathcal{L}}(c)|$ is polynomially bounded w.r.t. combined complexity; and (*iii*) deciding whether a set of $\mathcal{L}$-constraints S contains a clash can be done in polynomial space w.r.t. combined complexity. Condition (*i*) is required to guarantee termination, condition (*ii*) is needed to avoid that *e.g.* in the ($\sqcup_{\not\preceq}$) too many conjuncts are generated, while condition (*iii*) is needed to guarantee that not too much computational resources are required to decide whether a constraint set is clash-free or not. We call such lattices *ps-safe* (all lattices we have seen, except $\mathcal{L}'$, are ps-safe).

Proposition 4. *The satisfiability is PSPACE-complete w.r.t. combined complexity of a ps-safe lattice.*

Proof. [Sketch] We have seen that termination of the above algorithm is guaranteed. PSPACE-hardness follows directly from the PSPACE-completeness of the satisfiability problem in $\mathcal{ALC}$ and from Proposition 2. As for $\mathcal{ALC}$, our algorithm, as it is, requires exponential space due a well know problem: indeed any completion of $S = \{\langle x{:}C \succeq c \rangle\}$, where $C = (\exists R.A_{11}) \sqcap (\exists R.A_{12}) \sqcap \forall R.((\exists R.A_{21}) \sqcap (\exists R.A_{22}) \sqcap \ldots \forall R.((\exists R.A_{n1}) \sqcap (\exists R.A_{n2}))\ldots)$ contains at least $2^n + 1$ variables. Like in[20], we need to introduce so-called *trace rules*: *e.g.* the correspondent trace rule of the ($\exists_{\succeq}$) rule is

$$S \rightarrow_{T\exists_{\succeq}} S \cup \{\langle (a,b){:}R \succeq c \rangle, \langle b{:}C \succeq c \rangle\}$$
    if $\langle a{:}\exists R.C \succeq c \rangle \in S$, b new individual and no $\langle (a,b'){:}R' \succeq c' \rangle$ is in the current constraint set.

The trace rules relative to the rules ($\exists_{\not\preceq}$), ($\forall_{\preceq}$) and ($\forall_{\not\succeq}$) are similar. Assigning priority to all other rules, it can be shown that (*i*) a set of constraints S is satisfiable

iff no trace S' of S contains a clash; and (ii) the size of a trace S' of S is bounded polynomially by $|S| + |\mathcal{L}|$, and, thus, the satisfiability problem is in PSPACE.   □

The above result says us that no additional computational cost has to be paid for the major expressive power (if $\mathcal{L}$ is ps-safe, of course).

Concerning the BCVB problem, we may have a similar algorithm as for fuzzy $\mathcal{ALC}$[25]. In it, it has been shown that in $\mathcal{L}_{[0,1]}$, $glb(\Sigma, \alpha) \in N^{\Sigma}$ and $lub(\Sigma, \alpha) \in 1 - N^{\Sigma}$, where $N^{\Sigma} = \{0, 0.5, 1\} \cup \{c : \langle \alpha \geq c \rangle \in \Sigma\} \cup \{1 - c : \langle \alpha \leq c \rangle \in \Sigma\}$ and $1 - N^{\Sigma} = \{1 - c : c \in N^{\Sigma}\}$. Therefore, after determining N^{Σ} (complexity $O(|\Sigma|)$, $|N^{\Sigma}| \leq |\Sigma|$) and ordering N^{Σ} (complexity $O(|N^{\Sigma}| \log |N^{\Sigma}|)$), we can compute the glb by means of $O(\log |N^{\Sigma}|)$ entailment tests.

It is easily verified that the above method works for any total ordered lattice as well (we have to consider the certainty values appearing in the knowledge base only). In general, this is not true, as $e.g.$ a disjunction $\langle a:C_1 \sqcup C_2 \succeq c \rangle \in S$ may require to take into account the values in $D_{\mathcal{L}}(c)$ as well. It is still an open problem whether we can find, for any safe lattice $\mathcal{L}$ and a $\mathcal{L}$-KB Σ, a set of certainty values N^{Σ} such that $|N^{\Sigma}|$ is polynomially bounded by $|\Sigma| + |\mathcal{L}|$ and $glb(\Sigma, \alpha) \in N^{\Sigma}$.

Anyway, for any ps-safe lattice $\mathcal{L} = \langle T, \preceq \rangle$ we still may apply a simple iterative approximation algorithm for determining the glb:

(1) let $T_0 = T$ and $\bar{c} := f$.
(2) For $j = 0, 1, 2, \ldots$

    (a) take a value $c \in T_j$; if there is no such value then set $glb(\Sigma, \alpha) := \bar{c}$ and exit;
    (b) if $\Sigma \models_{\mathcal{L}} \langle \alpha \succeq c \rangle$ then set $\bar{c} := c$ and $T_{j+1} = T_j \setminus (\{c' : c' \not\succeq \bar{c}\} \cup \{c\})$, otherwise $T_{j+1} = T_j \setminus \{c' : c' \succeq c\}$;
    (c) go to step (2a)

Of course, the speed of convergence depends on the 'goodness' of the chosen value c in step (2a). Then algorithm for the lub is similar.

5. Transformation to classical DLs

In[27] it has been shown that reasoning in fuzzy $\mathcal{ALC}$ can be reduced to reasoning in classical DLs and, thus, already existing reasoners can be applied directly. We show that if the lattice is a linear order, then the same transformation process can be applied in our context as well.

For our purpose, we need to extend $\mathcal{ALC}$ to the DL $\mathcal{ALCH}$, which is $\mathcal{ALC}$ where terminological axioms are of the general form $C \sqsubseteq D$ (C, D concepts) and role axioms of the form $R \sqsubseteq R'$ are allowed. Now, let us assume that $\mathcal{L} = \langle T, \leq \rangle$ is a linear order. The idea is the following. Expressions of the form, $e.g.$, $\langle a:A \succeq c \rangle$ are translated into classical assertions of the form $a:A_{\succeq c}$, where $A_{\succeq c}$ is a new primitive concept. Informally, the concept $A_{\succeq c}$ represents the set of individuals, which are instance of A with degree $\succeq c$. The argument for $a:A_{\prec c}$ is similar. Of course, we

have to consider also the appropriate relationships among the introduced concepts. For instance, if $c \succeq c'$ we need the terminological axiom $A_{\succeq c} \sqsubseteq A_{\succeq c'}$.

Consider a $\mathcal{L}\text{-}\mathcal{ALC}$ KB Σ (w.l.o.g. we can assume it to be purely assertional). Define $X^{\Sigma} = \{f, t\} \cup \{c : \langle \alpha \succeq c \rangle \in \Sigma\} \cup \{\neg c : \langle \alpha \preceq c \rangle \in \Sigma\}$, from which we define $N^{\Sigma} = X^{\Sigma} \cup \{\neg c : c \in X^{\Sigma}\}$. If there is $\bar{c} \in \mathcal{T}$ such that $\neg \bar{c} = \bar{c}$, then we add $\bar{c}$ to X^{Σ}. We may assume that $N^{\Sigma} = \{c_1, \ldots, c_{|N^{\Sigma}|}\}$ and $c_i \prec c_{i+1}$, for $1 \leq i \leq |N^{\Sigma}| - 1$. Note that $c_1 = f$ and $c_{|N^{\Sigma}|} = t$. For each $c \in N^{\Sigma}$, for each relation $\bowtie \in \{\succeq, \succ, \preceq, \prec\}$, for each primitive concept A and role R occurring in Σ, consider a new primitive concept $A_{\bowtie c}$ and new role $R_{\succeq c}$ and $R_{\succ c}$, but we do not consider $A_{\prec f}, A_{\succ t}$ and $R_{\succ t}$ (which are not needed). Of course, we have to reflect the $\mathcal{L}$ order on the new concepts. This can easily be done by introducing axioms of the form

$$A_{\succeq c_{i+1}} \sqsubseteq A_{\succ c_i} \qquad\qquad A_{\succ c_i} \sqsubseteq A_{\succeq c_i}$$
$$A_{\prec c_j} \sqsubseteq A_{\preceq c_j} \qquad\qquad A_{\preceq c_i} \sqsubseteq A_{\prec c_{i+1}}$$

$$A_{\succeq c_j} \sqcap A_{\prec c_j} \sqsubseteq \bot \qquad\qquad A_{\succ c_i} \sqcap A_{\preceq c_i} \sqsubseteq \bot \qquad (3)$$

$$\top \sqsubseteq A_{\succeq c_j} \sqcup A_{\prec c_j} \qquad\qquad \top \sqsubseteq A_{\succ c_i} \sqcup A_{\preceq c_i} \ .$$

The terminological axioms relating the newly introduced roles are quite similar to the above axioms:

$$R_{\succeq c_{i+1}} \sqsubseteq R_{\succ c_i} \qquad R_{\succ c_i} \sqsubseteq R_{\succeq c_i} \ . \qquad (4)$$

We proceed now with the mapping of the $\mathcal{L}$-assertions in into crisp $\mathcal{ALCH}$ assertions. We define two mappings σ and ρ, defined as follows. Let ψ be a $\mathcal{L}$-assertion. Then σ maps a $\mathcal{L}$-assertion into a classical $\mathcal{ALCH}$ assertion, using ρ, as follows. In the following, we assume that $c \in \mathcal{T}$ and $\bowtie \in \{\succeq, \succ, \preceq, \prec\}$.

$$\sigma(\psi) = \begin{cases} a{:}\rho(C, \bowtie c) & \text{if } \psi = \langle a{:}C \bowtie c \rangle \\ (a, b){:}\rho(R, \bowtie c) & \text{if } \psi = \langle (a, b){:}R \bowtie c \rangle \ . \end{cases}$$

The mapping ρ encodes the idea we have previously presented and is inductively defined on the structure of concepts and roles. For roles, we have simply $\rho(R, \bowtie c) = R_{\bowtie c}$. So, for instance the $\mathcal{L}$-assertion $\langle (a, b){:}R \succeq c \rangle$ is mapped into the $\mathcal{ALCH}$ assertion $(a, b){:}R_{\succeq c}$. For concepts, we have the following inductive definitions (we give just some examples, the other cases are similar):

- for $\top$, *e.g.* $\rho(\top, \preceq c) = \bot$ if $c \prec 1$

- the case $\bot$ is similar to the case $\top$

- fort primitive concept A, $\rho(A, \bowtie c) = A_{\bowtie c}$

- for concept conjunction $C \sqcap D$, $\rho(C \sqcap D, \succeq c) = \rho(C, \succeq c) \sqcap \rho(D, \succeq c)$, $\rho(C \sqcap D, \preceq c) = \rho(C, \succeq c) \sqcup \rho(D, \succeq c)$

- the case of concept disjunction $C \sqcup D$ is dual to conjunction

- for concept negation $\neg C$, $\rho(\neg C, \succeq c) = \rho(C, \preceq \neg c)$

- for existential quantification $\exists R.C$, $\rho(\exists R.C, \succeq c) = \exists \rho(R, \succeq c).\rho(C, \succeq c)$ and $\rho(\exists R.C, \preceq c) = \forall \rho(R, \succ c).\rho(C, \preceq c)$

- for universal quantification $\forall R.C$, $\rho(\forall R.C, \succeq c) = \forall \rho(R, \succ \neg c).\rho(C, \succeq c)$ and $\rho(\forall R.C, \preceq c) = \exists \rho(R, \succeq \neg c).\rho(C, \preceq c)$.

The whole transformation process is polynomial in the size of Σ and satisfiability preserving.

Proposition 5. *A linear order $\mathcal{L}$-$\mathcal{ALC}$ KB is satisfiable iff its transformation into a crisp $\mathcal{ALCH}$ KB is satisfiable.*

Therefore, the entailment problem in linear order $\mathcal{L}$-$\mathcal{ALC}$ can be reduced to $\mathcal{ALCH}$. Finally, concerning the BTVB problem, we can apply a similar algorithm as for the non crisp case, except we use an oracle for classical entailment and the set of certainty values is not the whole $\mathcal{L}$, but just N^Σ.

6. Conclusion

We have presented a DL framework based on certainty lattices. Our main feature is that a sentence is not just true or false like in classical DLs, but *certain* to some degree, where the certainty value is taken from a certainty lattice. Syntax, semantics and a sound and complete tableau algorithm for reasoning in it has been presented. The complexity results shows that the additional expressive power has no impact from a computational complexity point of view, under plausible assumptions over a certainty lattice. This is especially important as the nice trade-off between computational complexity and expressive power of DLs contributes to their popularity. We have also shown that in case of linearly ordered lattices, a translation into classical DLs can be given.

While a calculus has been provided in this paper, it still remains an open issue whether to implement a reasoner from scratch or to take advantage of already existing DL reasoners, like RACER or FACT, *i.e.* to provide a translation of $\mathcal{L}$-$\mathcal{ALC}$ into a DL for more cases, or to rely on a many-valued first order reasoner, like 3TAP[g].

References

1. Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
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. P. Bonatti and A. Tettamanzi. Some complexity results on fuzzy description logics. In A. Petrosino V. Di Gesù, F. Masulli, editor, *WILF 2003 International Workshop on Fuzzy Logic and Applications*, LNCS 2955, Berlin, 2004. Springer Verlag.
4. T. Rosalba Giugno and Thomas Lukasiewicz. P-SHOQ(D): A probabilistic extension of SHOQ(D) for probabilistic ontologies in the semantic web. In *Proceedings of the 8th European Conference on Logics in Artificial Intelligence (JELIA'02)*, number 2424 in Lecture Notes in Artificial Intelligence, pages –. Springer-Verlag, 2002.
5. Jochen Heinsohn. Probabilistic description logics. In R. Lopez de Mantara and D. Pool, editors, *Proceedings of the 10th Conference on Uncertainty in Artificila Intelligence*, pages 311–318, 1994.
6. Steffen Hölldobler, Hans-Peter Störr, and Tran Dinh Khang. The subsumption problem of the fuzzy description logic ALC_{FH}. In *Proceedings of the 10th International Conference on Information Processing and Managment of Uncertainty in Knowledge-Based Systems, (IPMU-04)*, 2004.

---

[g]http://i12www.ira.uka.de/~threetap

7. Bernhard Hollunder. An alternative proof method for possibilistic logic and its application to terminological logics. In *10th Annual Conference on Uncertainty in Artificial Intelligence*, pages 327–335, Seattle, Washington, 1994. Morgan Kaufmann.

8. Manfred Jäger. Probabilistic reasoning in terminological logics. In *Proceedings of KR-94, 5-th International Conference on Principles of Knowledge Representation and Reasoning*, pages 305–316, Bonn, FRG, 1994.

9. Daphne Koller, Alon Levy, and Avi Pfeffer. P-CLASSIC: A tractable probabilistic description logic. In *Proc. of the 14th Nat. Conf. on Artificial Intelligence (AAAI-97)*, pages 390–397, 1997.

10. Laks V.S. Lakshmanan and Nematollaah Shiri. A parametric approach to deductive databases with uncertainty. *IEEE Transactions on Knowledge and Data Engineering*, 13(4):554–570, 2001.

11. Thomas Lukasiewicz. Fixpoint characterizations for many-valued disjunctive logic programs with probabilistic semantics. In *In Proceedings of the 6th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR-01)*, number 2173 in Lecture Notes in Artificial Intelligence, pages 336–350. Springer-Verlag, 2001.

12. C. Lutz, F. Wolter, and M. Zakharyaschev. A tableau algorithm for reasoning about concepts and similarity. In *Proceedings of the Twelfth International Conference on Automated Reasoning with Analytic Tableaux and Related Methods TABLEAUX 2003*, number 2796 in LNAI, Rome,Italy, 2003. Springer.

13. Carlo Meghini, Fabrizio Sebastiani, and Umberto Straccia. A model of multimedia information retrieval. *Journal of the ACM*, 48(5):909–970, 2001.

14. Bernhard Nebel. *Reasoning and revision in hybrid representation systems*. Springer, Heidelberg, FRG, 1990.

15. 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.

16. Peter F. Patel-Schneider. Adding number restrictions to a four-valued terminological logic. In *Proc. of the 7th Nat. Conf. on Artificial Intelligence (AAAI-88)*, pages 485–490, 1988.

17. Peter F. Patel-Schneider. A four-valued semantics for terminological logics. *Artificial Intelligence*, 38:319–351, 1989.

18. Dan Roth. On the hardness of approximate reasoning. *Artificial Intelligence Journal*, 82:273–302, 1996.

19. D. Sánchez and G.B. Tettamanzi. Generalizing quantification in fuzzy description logics. In *Proceedings 8th Fuzzy Days in Dortmund*, 2004.

20. Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48:1–26, 1991.

21. 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.

22. Umberto Straccia. A sequent calculus for reasoning in four-valued description logics. In *Proc. of the Int. Conf. on Analytic Tableaux and Related Methods (TABLEAUX-97)*, number 1227 in Lecture Notes in Artificial Intelligence, pages 343–357, Pont-à-Mousson, France, 1997.

23. Umberto Straccia. A fuzzy description logic. In *Proc. of the 15th Nat. Conf. on Artificial Intelligence (AAAI-98)*, pages 594–599, Madison, USA, 1998.

24. Umberto Straccia. A framework for the retrieval of multimedia objects based on four-valued fuzzy description logics. In F. Crestani and Gabriella Pasi, editors, *Soft Com-*

puting in Information Retrieval: Techniques and Applications, pages 332–357. Physica Verlag (Springer Verlag), Heidelberg, Germany, 2000.

25. Umberto Straccia. Reasoning within fuzzy description logics. *Journal of Artificial Intelligence Research*, 14:137–166, 2001.

26. Umberto Straccia. Towards a fuzzy description logic for the semantic web. Technical Report 2004-TR-64, Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, Pisa, Italy, 2004.

27. Umberto Straccia. Transforming fuzzy description logics into classical description logics. In *Proceedings of the 9th European Conference on Logics in Artificial Intelligence (JELIA-04)*, number 3229 in Lecture Notes in Computer Science, pages 385–399, Lisbon, Portugal, 2004. Springer Verlag.

28. Umberto Straccia. Uncertainty in description logics: a lattice-based approach. In *Proceedings of the 10th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, (IPMU-04)*, pages 251–258, 2004.

29. C. Tresp and R. Molitor. A description logic for vague knowledge. In *Proc. of the 13th European Conf. on Artificial Intelligence (ECAI-98)*, Brighton (England), August 1998.

30. John Yen. Generalizing term subsumption languages to fuzzy logic. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 472–477, Sydney, Australia, 1991.