Abstract

We present a fuzzy version of description logics with concrete domains. Main features are: (i) concept constructors are based on t-norm, t-conorm, negation and implication; (ii) concrete domains are fuzzy sets; (iii) fuzzy modifiers are allowed; and (iv) the reasoning algorithm is based on a mixture of completion rules and bounded mixed integer programming.

1 INTRODUCTION

In the last decade a substantial amount of work has been carried out in the context of Description Logics (DLs) [1]. Nowadays, DLs have gained even more popularity due to their application in the context of the Semantic Web [7]. Ontologies play a key role in the Semantic Web. An ontology consists of a hierarchical description of important concepts in a particular domain, along with the description of the properties (of the instances) of each concept. Web content is then annotated by relying on the concepts defined in a specific domain ontology. DLs play a particular role in this context as they are essentially the theoretical counterpart of the Web Ontology Language OWL DL, a state of the art language to specify ontologies.

However, OWL DL becomes less suitable in domains in which the concepts to be represented have not a precise definition. As we have to deal with Web content, it is easily verified that this scenario is, unfortunately, likely the rule rather than an exception. For instance, just consider the case we would like to build an ontology about flowers. Then we may encounter the problem of representing concepts like “Candia is a creamy white rose with dark pink edges to the petals”, “Jacaranda is a hot pink rose”, “Calla is a very large, long white flower on thick stalks”. As it becomes apparent such concepts hardly can be encoded into OWL DL, as they involve so-called fuzzy or vague concepts, like “creamy”, “dark”, “hot”, “large” and “thick”, for which a clear and precise definition is not possible.

The problem to deal imprecision has been addressed several decades ago by Zadeh ([20]), which gave birth in the meanwhile to the so-called fuzzy set and fuzzy logic theory. Unfortunately, despite the popularity of fuzzy set theory, relative little work has been carried out involving fuzzy DLs [5, 6, 10, 13, 15, 16, 18, 19].

Towards the management of vague concepts, we present a fuzzy extension of $\mathcal{ALC}(D)$ (the basic DL $\mathcal{ALC}$ [14] extended with concrete domains [9]). Main features are: (i) concept constructors are interpreted as t-norm, t-conorm, negation and implication. Current approaches consider conjunction as min, disjunction as max, negation as $1-x$ only. Given the important role norm based connectives have in fuzzy logic, a generalization towards this directions is, thus, desirable; (ii) concrete domains are fuzzy sets. This has not been addressed yet in the literature and is a natural way to incorporate vague concepts with explicit membership functions into the language. This requirement has already been pointed out by Yen in [19], but not yet taken into account formally; (iii) fuzzy modifiers are allowed, similarly to [18, 6]; and (iv) reasoning is based on a mixture of completion rules and bounded Mixed Integer Programming (bMIP). The use of bMIP in our context is novel and allows for effective implementations. Fuzzy $\mathcal{ALC}(D)$ enhances current approaches to fuzzy DLs and is in line with [17], in which the need of a fuzzy extension of DLs in the context of the Semantic Web has been highlighted. In it, a fuzzy version of OWL DL has been presented without a calculus. Our work is a step forward in this direction, as it presents a calculus for an important sub-language of OWL DL. We also show that the computation is more complicated than the classical counterpart due to the generality of the connectives.

We proceed as follows. The following section presents fuzzy $\mathcal{ALC}(D)$. Section 3 presents the reasoning pro-
Fuzzy sets [20] allow to deal with vague concepts like low pressure, high speed and the like. A fuzzy set $A$ with respect to a universe $X$ is characterized by a membership function $\mu_A : X \rightarrow [0, 1]$, or simply $A(x) \in [0, 1]$, assigning an $A$-membership degree, $A(x)$, to each element $x$ in $X$. $A(x)$ gives us an estimation of the belonging of $x$ to $A$. In fuzzy logics, the degree of membership $A(x)$ is regarded as the degree of truth of the statement “$x$ is $A$”. Accordingly, in our fuzzy DL, a concept $C$ will be interpreted as a fuzzy set and, thus, concepts become imprecise; and, consequently, e.g. the statement “$a$ is an instance of concept $C$”, will have a truth-value in $[0, 1]$ given by the membership degree $C(a)$.

**Syntax.** Recall that $\text{ALC}(\mathbb{D})$ is the basic DL $\text{ALC}$ [14] extended with concrete domains [9] allowing to deal with data types such as strings and integers. In fuzzy $\text{ALC}(\mathbb{D})$, however, concrete domains are fuzzy sets. A fuzzy concrete domain (or simply fuzzy domain) is a pair $(\Delta_0, \Phi_0)$, where $\Delta_0$ is an interpretation domain and $\Phi_0$ is the set of fuzzy domain predicates $d$ with a predefined arity $n$ and an interpretation $d^\mathbb{D} : \Delta_0^n \rightarrow [0, 1]$, which is an $n$-ary fuzzy relation over $\Delta_0$. To the ease of presentation, we assume the fuzzy predicates have arity one, the domain is a subset of the rational numbers $\mathbb{Q}$ and the range is $[0, 1] \cap \mathbb{Q}$ (in the following, whenever we write $[0, 1]$, we mean $[0, 1] \cap \mathbb{Q}$). For instance, we may define the predicate $\leq_{18}$ as an unary crisp predicate over the natural numbers denoting the set of integers smaller or equal to 18, i.e.

$$\leq_{18}(x) = \begin{cases} 1 & \text{if } x \leq 18 \\ 0 & \text{otherwise} \end{cases}.$$ 

On the other hand, $\text{Young}$ may be a fuzzy domain predicate denoting the degree of youngness of a person’s age with definition

$$\text{Young}(x) = \begin{cases} 1 & \text{if } x \leq 10 \\ (30 - x)/20 & \text{if } 10 \leq x \leq 30 \\ 0 & \text{if } x \geq 30 \end{cases}.$$

Concerning fuzzy domain predicates, we recall that in fuzzy set theory and practice there are many membership functions for fuzzy sets membership specification. However, the trapezoidal $\text{tr}_{z}(x; a, b, c, d)$, the triangular $\text{tri}(x; a, b, c, d)$, the $L$-function (left shoulder function) $L(x; a, b)$ and the $R$-function (right shoulder function) $R(x; a, b)$ are simple, yet most frequently used to specify membership degrees (see Figure 1). Note that $\text{tri}(x; a, b, c) = \text{tr}_{z}(x; a, b, b, c)$. Also, we have that $\text{Young}(x) = L(x; 10, 30)$ holds.

We also consider fuzzy modifiers in fuzzy $\text{ALC}(\mathbb{D})$. Fuzzy modifiers, like very, more or less and slightly, apply to fuzzy sets to change their membership function. Formally, a modifier is a function $f_m : [0, 1] \rightarrow [0, 1]$. For instance, we may define $\text{very}(x) = x^2$, while define $\text{slightly}(x) = \sqrt{x}$. Modifiers have been considered, for instance, in [6, 18].

Now, let $C$, $R_a$, $R_c$, $I_a$, $I_c$ and $M$ be non-empty finite and pair-wise disjoint sets of concepts names (denoted $A$), abstract roles names (denoted $R$), concrete roles names (denoted $T$), abstract individual names (denoted $a$), concrete individual names (denoted $c$) and modifiers (denoted $m$). $R_a$ contains a non-empty subset $F_a$ of abstract feature names (denoted $r$), while $R_c$ contains a non-empty subset $F_c$ of concrete feature names (denoted $t$). Features are functional roles. The set of fuzzy $\text{ALC}(\mathbb{D})$ concepts is defined by the following syntactic rules ($d$ is a unary fuzzy domain predicate):

$$C \rightarrow T \mid \bot \mid A \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \neg C \mid \forall R.C \mid \exists R.C \mid \forall T.D \mid \exists T.D \mid m(C)$$

$$D \rightarrow d \mid \neg d$$

A TBox $T$ consists of a finite set of terminological axioms of the form $A \sqsubseteq C$ ($A$ is sub-concept of $C$) or $A = C$ ($A$ is defined as the concept $C$), where $A$ is a concept name and $C$ is concept. We also assume that no concept $A$ appears more than once on the left hand side of a terminological axiom and that no cyclic definitions are present in $T$. 1 Note that in classical DLs, terminological axioms are of the form $C \sqsubseteq D$, where $C$ and $D$ are concepts. While from a semantics point of view it is easy to consider them as well (see [17]), we have not yet found a calculus to deal with such axioms. Using axioms we may define the concept of a minor as

$$\text{Minor} = \text{Person} \sqcap \exists \text{Age.} \leq_{18}$$

while

$$\text{YoungPerson} = \text{Person} \sqcap \exists \text{Age.} \text{Young}$$

will denote a young person. Similarly, we may represent “Callas a very large, long white flower on thick stalks” as $\text{Calla} = \text{Flower} \sqcap \exists \text{R.A}.$, which may generate an infinite computation in the model generation phase (assume that $A$ has an instance). For classical DLs, clever blocking conditions has been developed, which however do not exists yet for the fuzzy case.
the chosen $t$-norm, we have that then, according to the chosen t-norm, we have that $x \Rightarrow y = \ominus_L x \oplus_L y$. The same holds using Kleene-Dienes implication, Łukasiewicz negation and Gödel $t$-conorm. On the other hand $x \Rightarrow_P y \neq \ominus_G x \oplus_P y$. We conclude the discussion on fuzzy implication by noting that we have the following inferences: assume $x \geq n$ and $x \Rightarrow y \geq m$. Then (i) under Kleene-Dienes implication we infer that if $n > 1 - m$ then $y \geq m$ (this is used in [15]). (ii) under residuum based implication w.r.t. a $t$-norm $\otimes$, we infer that $y \geq n \otimes m$, which we will use in this paper.

The semantics of fuzzy $\mathcal{ALC}$ is as follows. A fuzzy interpretation $\mathcal{I}$ with respect to a concrete domain $D$ is a pair $\mathcal{I} = (\Delta^D, \mathcal{I})$ consisting of a non empty set $\Delta^D$ (called the domain), disjoint from $\Delta_0$, and of a fuzzy interpretation function $\mathcal{I}$ that assigns (i) to each abstract concept $C \in \mathcal{C}$ a function $C^\mathcal{I}: \Delta^D \rightarrow [0,1]$; (ii) to each abstract role $R \in \mathcal{R}$, a function $R^\mathcal{I}: \Delta^D \times \Delta^D \rightarrow [0,1]$; (iii) to each abstract feature $r \in \mathcal{F}_a$ a partial function $r^\mathcal{I}: \Delta^D \times \Delta^D \rightarrow [0,1]$ such that for all $u \in \Delta^D$ there is an unique $w \in \Delta^D$ on which $r^\mathcal{I}(u, w)$ is defined; (iv) to each abstract individual $a \in \mathcal{I}_a$ an element in $\Delta^D$; (v) to each concrete individual $c \in \mathcal{I}_c$ an element in $\Delta_0$; (vi) to each concrete role $R \in \mathcal{R}_c$ a function $R^\mathcal{I}: \Delta^D \times \Delta_0 \rightarrow [0,1]$; (vii) to each concrete feature $t \in \mathcal{F}_a$, a partial function $t^\mathcal{I}: \Delta^D \times \Delta_0 \rightarrow [0,1]$ such that for all $u \in \Delta^D$ there is an unique $v \in \Delta_0$ on which $t^\mathcal{I}(u, o)$ is defined; (viii) to each modifier $m \in \mathcal{M}$ the function $m^\mathcal{I}: [0,1] \rightarrow [0,1]$; (ix) to each unary concrete predicate $d$ the fuzzy relation $d^\mathcal{I}$; $\Delta_0 \rightarrow [0,1]$ and to $\neg d$ the negation of $d^\mathcal{I}$. The mapping $\mathcal{I}$ is extended to concepts and roles as follows (where $u \in \Delta^D$): $\top^\mathcal{I}(u) = 1$, $\perp^\mathcal{I}(u) = 0$.

\[
\begin{align*}
(C_1 \sqcap C_2)^\mathcal{I}(u) &= C_1^\mathcal{I}(u) \sqcap C_2^\mathcal{I}(u) \\
(C_1 \sqcup C_2)^\mathcal{I}(u) &= C_1^\mathcal{I}(u) \sqcup C_2^\mathcal{I}(u) \\
(\neg C)^\mathcal{I}(u) &= \ominus C^\mathcal{I}(u) \\
(m(C))^\mathcal{I}(u) &= f_m(C^\mathcal{I}(u)) \\
(\forall R.C)^\mathcal{I}(u) &= \inf_{w \in \Delta^D} R^\mathcal{I}(u, w) \Rightarrow C^\mathcal{I}(w) \\
(\exists R.C)^\mathcal{I}(u) &= \sup_{w \in \Delta^D} R^\mathcal{I}(u, w) \sqcap C^\mathcal{I}(w) \\
(\forall T.D)^\mathcal{I}(u) &= \inf_{o \in \Delta_0} T^\mathcal{I}(u, o) \Rightarrow D^\mathcal{I}(o) \\
(\exists T.D)^\mathcal{I}(u) &= \sup_{o \in \Delta_0} T^\mathcal{I}(u, o) \sqcap D^\mathcal{I}(o) .
\end{align*}
\]

The mapping $\mathcal{I}$ is extended to assertion axioms as follows (where $a, b \in \mathcal{I}_a$): $(a: C)^\mathcal{I} = C^\mathcal{I}(a^\mathcal{I})$ and $((a, b): R)^\mathcal{I} = R^\mathcal{I}(a^\mathcal{I}, b^\mathcal{I})$. The notion of satisfiability...
of a fuzzy axiom \( E \) by a fuzzy interpretation \( I \), denoted \( I \models E \), is defined as follows: \( I \models A \subseteq C \) iff for all \( u \in \Delta^I \), \( A^I(u) \leq C^I(u) \) (this definition is equivalent to \( \inf_{u \in \Delta^I} A^I(u) \Rightarrow C^I(u) \) = 1, which is derived directly from its FOL translation \( \forall x. A(x) \Rightarrow C(x) \); 
\( I \models (\alpha, n) \) iff \( \alpha^I \geq n \); 
\( I \models (\alpha, n) \) iff \( \alpha^I = b^I \); and 
\( I \models (\alpha, n) \) iff \( \alpha^I \neq b^I \). The notion of satisfiability (is model) of a knowledge base \( K = (T, A) \) and entailment of an assertional axiom is straightforward. Concerning terminological axioms, we also introduce degrees of subsumption. We say that \( K \) entails \( A \subseteq B \) to degree \( n \in [0, 1] \), denoted \( K \models A \sqsubseteq B, n \) iff for every model \( I \) of \( K \), \( \inf_{u \in \Delta^I} A^I(u) \Rightarrow B^I(u) \) \( \geq n \).

**Example 1** Consider the following simplified excerpt of a knowledge base about cars:

\[
\text{SportsCar} = \exists \text{speed}. \text{very}(\text{High}), \\
\langle \text{mg_mgb:} \text{speed} \leq 170, 0.1 \rangle, \\
\langle \text{ferrari_enzo:} \text{speed} \geq 350, 0.1 \rangle, \\
\langle \text{audi_tt:} \text{speed} = 243, 1 \rangle.
\]

*speed* is a concrete feature. The fuzzy domain predicate *High* has membership function \( \text{High}(x) = R(x; 80, 250) \). It can be shown that

\[
K \models \langle \text{mg_mgb:} \neg \text{SportsCar}, 0.72 \rangle \\
K \models \langle \text{ferrari_enzo:} \text{SportsCar}, 1 \rangle \\
K \models \langle \text{audi_tt:} \text{SportsCar}, 0.92 \rangle.
\]

Note how the maximal speed limit of the *mg_mgb* car \((\leq 170)\) induces an upper limit, 0.28 = 1 − 0.72, on the membership degree of being *mg_mgb* a *SportsCar*.

**Example 2** Consider \( K \) with terminological axioms (1) and (2). Then under Zadeh logic \( K \models \langle \text{Minor} \sqsubseteq \text{YoungPerson}, 0.5 \rangle \) holds (see Example 3).

Finally, given \( K \) and an axiom \( \alpha \), it is of interest to compute its best lower degree bound. The greatest lower bound of \( \alpha \) w.r.t. \( K \), denoted \( glb(K, \alpha) \), is 
\( glb(K, \alpha) = \sup \{ n; K \models (\alpha, n) \} \), where \( \sup \emptyset = 0 \). Determining the \( glb \) is called the Best Degree Bound (BDB) problem. For instance, the entailments in Examples 1 and 2 are the best possible degree bounds. Note that, \( K \models (\alpha, n) \) iff \( glb(K, \alpha) \geq n \). Therefore, the BDB problem is the major problem we have to consider in fuzzy ALC(D), which we address in the next section.

We finally point out that the expressions of a knowledge base should not necessarily be considered as the language to be presented to the user, but rather are the internal representation of a reasoning system. Indeed, it may be questionable whether a user may be able or should be allowed to express \( (\alpha, n) \) as she might be unsure which value to chose for \( n \). In case a user is not allowed to specify a truth value, alternative options might be: assume that \( K = (T, A) \) consists of a TBox \( T \), but where the ABox \( A \) is a set of (unweighted) assertions \( \alpha \) only. For \( n \in [0, 1] \), let \( (K, n) \) be the fuzzy knowledge base \( (T, (A, n)) \), where \( (A, n) \) is the ABox of fuzzy assertions \( \{ (\alpha, n) ; \alpha \in A \} \). Then alternative definitions of the BDB problem are, for instance, (i) \( glb(K, \alpha) = \sup \{ n; (K, 1) \models (\alpha, n) \} \); and (ii) \( glb(K, \alpha) = \sup \{ n; (K, n) \models (\alpha, n) \} \). Under the assumptions described below, these problems can still be solved by the method presented in the next section.

### 3 Reasoning in Fuzzy ALC(D)

To make our proof system for fuzzy ALC(D) correct and complete, we will assume that the chosen t-norm \( \odot \), t-conorm \( \oplus \), negation \( \neg \) and implication \( \Rightarrow \) are such that always \( x \odot y \equiv x \odot (x \odot y) \); \( x \Rightarrow y \equiv x \odot y \); and \( \odot \forall x. A(x) \equiv \exists x. A(x) \) hold for all fuzzy sets \( A \), where \( \forall \) is interpreted as \( \inf \) and \( \exists \) as sup. These are true, e.g. for Lukasiewicz logic, but not for Gödel logic. 3

Due to the restrictions on the chosen fuzzy functions, we do have that \( (\forall R.C)^T = (\neg \exists R.\neg C)^T \). This will allow us to transform concept expressions into a semantically equivalent Negation Normal Form (NNF), which is obtained by pushing in the usual manner negation on front of concept names, modifiers and concrete predicate names only. With \( \text{nnf}(C) \) we denote the NNF of concept \( C \).

Additionally, we assume the set of truth degrees in \([0, 1]\) we will deal with 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 truth degrees can be represented. In particular, this includes our case were we use the rational numbers in \([0, 1] \cap Q \) under a given fixed precision \( p \) of numbers a computer can work with.

The basic idea behind our reasoning algorithm is as follows. Consider \( K = (T, A) \). In order to solve the BDB problem, we combine appropriate DL completion rules with methods developed in the context of Many-Valued Logics (MVLs) [3]. In order to determine e.g. \( glb(K, \alpha; C) \), we consider an expression of the form \( \langle \alpha; C : \odot x \rangle \) (informally, \( \langle \alpha; C \leq x \rangle \)), where \( x \) is a \([0, 1]\)-valued variable. Then we construct a tableau for \( K = (T, A \cup \{ \langle \alpha; \neg C, \odot x \rangle \}) \) in which the application of satisfiability preserving rules generates new assertion axioms together with inequations over \([0, 1]\)-valued variables. These inequations have to hold in

---

3It is worth noting that (see [5]) the axiom \( T \sqsubseteq \neg (\forall R.A) \sqcap (\neg \exists R.\neg A) \) has no classical model. However, in [5] it is shown that in Gödel logic it has no finite model, but has an infinite model.
order to respect the semantics of the DL constructors. Finally, in order to determine the greatest lower bound, we minimize the original variable \( x \) such that all constraints are satisfied. In general, depending on the semantics of the DL constructors and fuzzy domain predicates we may end up with a general, bounded Non Linear Programming optimization problem. In this paper, however, we will limit the choice of the semantics of concept constructors, modifiers and fuzzy domain predicates in such a way that we end up with a bounded Mixed Integer Program (bMIP) optimization problem [12]. Interestingly, as for the MVL case, the tableaux rules can be designed in such a way that sound, complete and linear tableaux rules can be obtained from it. Also, using ideas from disjunctive programming, the tableaux rules can be designed in such a way that a one-branch tree only is generated. See [3] for more on this issue and on bMIP-representability conditions for connectives. For instance, classical logic, Zadeh’s fuzzy logic, and Lukasiewicz connectives, are bMIP-representable, while Gödel negation is not. In general, connectives whose graph can be represented as the union of a finite number of convex polyhedra are bMIP-representable [8], however, discontinuous functions may not be bMIP representable.

**The BDB problem.** We start with some preprocessing steps as for classical DLs [11]. First, each terminological axiom \( A \subseteq C \in T \) can be replaced with \( A = C \cap A^* \), where \( A^* \) is a new concept name. Let \( K' \) the obtained knowledge base. Second, the newly obtained \( K' \) can be expanded by substituting every concept name \( A \) occurring in \( K \), which is defined in \( T \), with its defining term in \( T \). Although, the expanded knowledge base may become of exponential size, the properties from a semantics point of view are left unchanged. Let \( K'' \) the obtained knowledge base. Finally, each concept occurring in \( K'' \) is then transformed into NNF. This last operations does not affect the semantics due to the restrictions we made on the fuzzy constructors. Notice that negation may appear on front of modifiers in the from \( \neg m(C) \), where \( C \) is a complex concept. Now, let \( V \) be a new alphabet of variables \( x \) ranging over \([0,1] \), \( W \) be a new alphabet of 0-1 variables \( y \). We extend fuzzy assertions to the form \( \langle \alpha, l \rangle \), where \( l \) is a linear expression over variables in \( V, W \) and real values. A linear constraint is of the form \( l \geq l' \) or \( l \leq l' \), where \( l, l' \) are linear expressions over variables in \( V, W \) and rational values. The satisfiability notion of linear constraints is immediate. A constraint set \( S \) is a set of terminological axioms, fuzzy assertion axioms, (in)equality axioms and linear constraints. \( I \) satisfies \( S \) iff \( I \) satisfies all elements of it. With \( S_0 \) we denote the constraint set \( S_0 = T \cup A \). We will see later how to determine the satisfiability of a constraint set.

In the following, we assume that \( S_0 \) is satisfiable, otherwise \( glb(K, \alpha) = 1 \). Note that our algorithm can be used to test the satisfiability of \( S_0 \) in the first place. As in [15], concerning fuzzy role assertions, we have that \( K = \langle (a,b): R, n \rangle \) iff \( \langle (a,b): R, m \rangle \in A \) with \( m \geq n \). Therefore, \( glb(K, \langle a,b \rangle; R) = \max \{ n: (R(a,b), n) \in A \} \). So we do not consider this case further. Now, let us determine \( glb(K, \langle a,c \rangle) \). As anticipated, \( glb(K, \langle a,c \rangle) \) is determined by the minimal value of \( x \) such that the constraint set \( S = S_0 \cup \{ \langle a : -C, \exists x \rangle \} \) is satisfiable. Similarly, for a terminological axiom \( A \subseteq B \), we can compute \( glb(K, A \subseteq B) \) as the minimal value of \( x \) such that the constraint set \( S = S_0 \cup \{ \langle a : A \cap -B, \exists x \rangle \} \) is satisfiable, where \( a \) is new abstract individual. Therefore, the BDB problem can be reduced to minimal satisfiability problem.

Note that we previously gave also alternative definitions of the BDB problem. These cases can be reduced to the satisfiability problem as well. Indeed, (i) for \( glb(K, \langle a,c \rangle) = \sup \{ n: (K, I) |= \langle a,c,n \rangle \} \), deter-
mine the minimal value of \( x \) such that the constraint set \( S = T \cup \{ a; \sim A, l \} \cup \{ a; \sim C, \circ x \} \) is satisfiable; while (ii) for \( g \ell b(K, a; C) = \sup \{ n; (K, n) \models (a; C, n) \} \), determine the minimal value of \( x \) such that the constraint set \( S = T \cup \{ a; x \} \cup \{ a; \sim C, \circ x \} \) is satisfiable. We will deserve to this issue more space in the extended version of this paper.

The Satisfiability problem. We assume that the concept constructors, concept modifiers and fuzzy domains predicates are bMIB representable (as e.g., the membership functions in Figure 1). In particular, we present a correct and complete proof system where the DL connectives are interpreted according to Zadeh logic, while modifiers and fuzzy domain predicates are specified as a combination of linear functions over \([0,1]\) and \(Q\), respectively, as specified in Appendix A. We also present a correct and complete proof system for Lukasiewicz logic in Appendix B.

Our satisfiability checking calculus is based on a set of constraint propagation rules transforming a set \( S \) of constraints into “simpler” satisfiability preserving constraint sets \( S_i \) until either \( S_i \) contains a clash or no rule can be further be applied to \( S_i \). If \( S_i \) contains a clash then \( S_i \) and, thus \( S \), is immediately not satisfiable. Otherwise, we apply a bMIP oracle to solve the set of linear constraints in \( S_i \) to determine either the satisfiability of the set or the minimal value for a given variable \( x \), making \( S_i \) satisfiable. We assume that a constraint set \( S \) is reflexive, symmetric and transitively closed concerning the equality axioms. \( S \) contains a clash iff either \( (a; \perp, n) \in S \) with \( n > 0 \), or \( \{ a \approx b, a \neq b \} \subseteq S \). The rules follow easily from the bMIP representations. Each rule instantiation is applied at most once. Before we can formulate the rules we need a technical definition involving feature roles (see [9]). Let \( S \) be a constraint set, \( r \) an abstract feature and both \( \{ (a, b_1); r, l_1 \} \) and \( \{ (a, b_2); r, l_2 \} \) occur in \( S \). Then we call such a pair a fork. As \( r \) is a function, such a fork means that \( b_1 \) and \( b_2 \) have to be interpreted as the same individual. A fork \( \{ (a, b_1); r, l_1 \}, \{ (a, b_2); r, l_2 \} \) can be deleted by replacing all occurrences of \( b_2 \) in \( S \) by \( b_1 \). A similar argument applies to concrete feature roles. At the beginning, we remove the forks from \( S_o \).

We assume that forks are eliminated as soon as they appear (as part of a rule application) with the proviso that newly generated individuals are replaced by older ones and not vice-versa. With \( x_\alpha \) we denote the variable associated to the atomic assertion \( \alpha \) of the form \( a; A \) or \( (a, b); R \). \( x_\alpha \) will take the truth value associated to \( \alpha \), while with \( x_c \) we denote the variable associated to the concrete individual \( c \). The rules are the following:

\( \textbf{R.A.} \) If \( (a, l) \in S_i \) and \( \alpha \) is an atomic assertion of the form \( a: A \) or \( (a, b); R \) then \( S_{i+1} = S_i \cup \{ x_\alpha : A \leq l - 1 \} \).

\( \textbf{R.B.} \) If \( (a; C \cap D, l) \in S_i \) then \( S_{i+1} = S_i \cup \{ (a; C, l) \}, (a; D, l) \} \).

\( \textbf{R.C.} \) If \( (a; C \cup D, l) \in S_i \) then \( S_{i+1} = S_i \cup \{ (a; C, x_1), (a; D, x_2), x_1 + x_2 = l, x_1 \leq y, x_2 \leq 1 - y, x_1 \in [0,1], y \in \{0,1\} \}, \) where \( x_1 \) is a new variable, \( y \) is a new control variable.

\( \textbf{R.D.} \) If \( (a; R.C, l) \in S_i \) then \( S_{i+1} = S_i \cup \{ (a; b, R), (b; C, l) \} \), where \( b \) is a new abstract individual. The case for concrete roles is similar.

\( \textbf{R.E.} \) If \( (a; \exists R.C, l) \in S_i \) then \( S_{i+1} = S_i \cup \{ (a; b, R), R, l \} \). The case for concrete roles is similar.

\( \textbf{R.M.} \) If \( (a; m(C), l) \in S_i \) then \( S_{i+1} = S_i \cup \gamma(a; C, l) \), where the set \( \gamma(a; C, l) \) is obtained from the bMIP representation (see appendix) of \( g(m) \) as follows: replace in \( g(m) \) all occurrences of \( x_2 \) with \( l \). Then resolve for \( x_1 \) and replace all occurrences of the form \( x_1 \geq l \) with \( (a; C, l') \), while replace all occurrences the form \( x_1 \leq l \) with \( \text{minf}(\sim C, 1 - l') \).

\( \textbf{R.m} \). The case \( (a; \sim m(C), l) \in S_i \) is similar to rule \( \textbf{R.m} \), where we use the bMIP representation of \( g(m) \) in place of \( g(m) \).

\( \textbf{R.d} \). If \( (c; d, l) \in S_i \) then \( S_{i+1} = S_i \cup \gamma(c; d, l) \), where the set \( \gamma(c; d, l) \) is obtained from the bMIP representation of \( g(d) \) by replacing all occurrences of \( x_2 \) with \( l \) and \( x_1 \) with \( x_c \).

\( \textbf{R.d} \). The case \( (c; \sim d, l) \in S_i \) is similar to rule \( \textbf{R.d} \), where we use the bMIP representation of \( g(d) \) in place of \( g(d) \).

Note that an unique branch is generated in the tableaux of \( S_0 \), though it can be of exponential length. Furthermore, let us comment the \( \textbf{R.l} \) rule. By reasoning by case, for \( y = 0 \), we have \( x_1 = 0, x_2 \leq 1, x_2 = l \), while for \( y = 1 \), we have \( x_2 = 0, x_1 \leq 1, x_1 = l \). Therefore, the control variable \( y \) simulates the two branchings of the disjunction. A similar argument applies to the other rules.

We say that a constraint set \( S' \) obtained from rule applications to \( S \) is a completion of \( S \) iff no more rule can be applied to \( S' \). The following can be shown.

**Proposition 1** Let \( S \) be a constraint set. The rules are satisfiability preserving and a completion of \( S \) is obtained after a finite number of rule applications.

**Proposition 2** Consider \( K(T, A) \) and let \( \alpha \) be a concept assertion axiom \( a: C \) or a terminological axiom \( a: A \) or \( (a, b); R \).
A ⊆ B. Then in finite time we can determine \( \text{glb}(K, \alpha) \) as the minimal value of \( x \) such that the completion of \( S = T \cup \mathcal{A}(\{\alpha', 1 - x\}) \) is satisfiable, where (i) \( \alpha' = a: \neg C \) if \( \alpha = a: C \), (ii) \( \alpha' = a: A \cap \neg B \) if \( \alpha = A \subseteq B \).

**Example 3** Let us consider a simplified version of Example 2, by showing that \( K \models (\text{Minor} \sqsubseteq \text{YoungPerson}, 0.6) \) holds, where Minor = \( \leq 18 \) and YoungPerson = Young, and that this is the best degree bound. We use \( M, Y \) and \( \text{YP} \) as a shorthand for Minor, YoungPerson and Young, respectively. For ease, a variable \( x_\alpha \), where \( \alpha \) is an assertion is simply written as \( \alpha \). We have to consider

\[
S_0 \cup \{\langle b; M \cap \neg \text{YP}, 1 - x \rangle\} ,
\]

where \( b \) is a new abstract individual. That is, we have to minimize \( x \) such that

\[
S_1 = T \cup \{\langle b; \leq 18 \cap \neg Y, 1 - x \rangle, x \in [0, 1]\}
\]

is satisfiable. By application of the \( R \cap \) rule we get

\[
S_2 = S_1 \cup \{\langle b; \leq 18, 1 - x \rangle, \langle b; \neg Y, 1 - x \rangle\} .
\]

By abuse of notation, we write \( \langle b; \neg Y, 1 - x \rangle \) as \( b; Y \leq x \). Now, for \( x = 1 \), \( S_2 \) is satisfiable, while for \( x = 0 \), from \( \langle b; \leq 18, 1 \rangle, 0 \leq x_b \leq 18 \) follows and from \( b; Y \leq 0 \), \( x_b \geq 30 \) is required and, thus, \( S_2 \) is not satisfiable (for \( x = 0 \)). For \( 0 < x < 1 \), \( 0 \leq x_c \leq 18 \) should hold. Furthermore, over \([0, 30]\) it can be shown that

\[
\bar{g}(Y) = \{(x_1, x_2): x_1 \leq 10 + 20y, x_2 \geq (1 - y), x_1 \geq 10 \}
\]

\[
\{x_1 \leq 30, x_1 + 20x_2 \geq 30 \}
\]

\( y \in [0, 1] \}

holds (see Equation 3 in the appendix). This means that, from \( S_2 \), by applying the \( R_d \) rule to \( b; Y \leq x \), we get the set \( S_3 = S_2 \cup \{x_3 \leq 10 + 20y, x \geq (1 - y), x_b \geq 10 \}
\]

\[
x_3 \geq 30, x_3 + 20x \geq 30, x \in [0, 1], y \in [0, 1]\}
\]

\( y \in [0, 1] \}

For \( y = 0 \), \( x_b \leq 10 \) and \( x = 1 \) have to hold and \( S_3 \) is still satisfiable. On the other hand, for \( y = 1 \), \( x_b \geq 10 \) and \( x_b + 20x \geq 30 \) hold. That is, \( x \geq (30 - x_b)/20 \). As \( 10 \leq x_b \leq 18 \), the minimal value of \( x \) satisfying \( S_3 \) under this condition is, thus, \( x = 3/5 \). Therefore, the minimal solution \( x \) satisfying \( S_3 \) is \( x = 3/5 \).

## 4 RELATED WORK

The first work on fuzzy DLs is due to Yen ([19]) who considered a sub-language of \( \mathcal{ALC} \), \( \mathcal{FL}^- \) [2]. However, it already informally talks about the use of modifiers and concrete domains. Though, the unique reasoning facility, the subsumption test, is a crisp yes/no question. Tresp ([18]) considered fuzzy \( \mathcal{ALC} \) extended with a special form of modifiers, which are a combination of two linear functions, as we described in the appendix.

min, max and \( 1 - x \) membership functions has been considered and a sound and complete reasoning algorithm testing the subsumption relationship has been presented. Similar to our approach, a linear programming oracle is needed. Assertional reasoning has been considered by Straccia ([15]), where fuzzy assertion axioms have been allowed in fuzzy \( \mathcal{ALC} \) (with min, max and \( 1 - x \) functions), concept modifiers are not allowed however. He also introduced the BDB problem and provided a sound and complete reasoning algorithm based on completion rules ([16]) provides a translation of fuzzy \( \mathcal{ALC} \) into classical \( \mathcal{ACC} \). For an application see [10]. In the same spirit [6] extend Straccia’s fuzzy \( \mathcal{ALC} \) with concept modifiers of the form \( f_m(x) = x^\beta \), where \( \beta > 0 \). A sound and complete reasoning algorithm for the graded subsumption problem, based on completion rules, is presented. [13] starts addressing the issue of alternative semantics of quantifiers in fuzzy \( \mathcal{ALC} \) (without the assertional component). No reasoning algorithm is given. Finally, [5] considers \( \mathcal{ACC} \) under arbitrary \( t \)-norm and reports, among others, a procedure deciding \( \models C \subseteq D \geq 1 \) and deciding whether \( C \subseteq D \geq 1 \) is satisfiable, by a reduction to the propositional BL logic.

## 5 CONCLUSIONS AND OUTLOOK

We have presented fuzzy \( \mathcal{ALC}(D) \) showing that its representation and reasoning capabilities go clearly beyond current approaches to fuzzy DLs. We believe that the fuzzy extension of \( \mathcal{ALC}(D) \) allows to express naturally a wide range of concepts of actual domains, for which a classical representation is unsatisfactory. Fuzzy \( \mathcal{ALC}(D) \) enhances current approaches as we allow arbitrary \( b \text{MIP} \)-representable concept constructors, modifiers and fuzzy domain predicates to appear in a \( \mathcal{ALC}(D) \) knowledge base. The entailment and the subsumption relationship hold to a certain degree. We also presented a solution to the BDB problem based on a minimization problem on \( b \text{MIP} \).

Future work involves the extension of fuzzy \( \mathcal{ALC}(D) \) to \( \mathcal{SHOIN}(D) \), the theoretical counterpart of OWL DL.

Another direction is in extending fuzzy DLs with fuzzy quantifiers, where \( \forall \) and \( \exists \) are replaced with fuzzy quantifiers like most, some, usually and the like (see [13] for a preliminary work in this direction). This allows to define concepts like

\[
\text{TopCust} = \text{Customer} \sqcap (\text{Usually} \text{buys}. \text{ExpensItem})
\]

\[
\text{ExpensItem} = \text{Item} \sqcap \exists \text{price}. \text{High} .
\]

Here, the fuzzy quantifier \text{Usually} replaces the classical quantifier \( \forall \) and \text{High} is a fuzzy concrete predicate. Fuzzy quantifiers can be applied to inclusion axioms as
well, allowing to express, e.g.

(Most)Bird ⊆ FlyingObject.

Here the fuzzy quantifier Most replaces the classical universal quantifier ∀ assumed in the inclusion axioms. The above axiom allows to state that most birds fly.

A ON MEMBERSHIP FUNCTIONS

As a building blocks for membership function specification, we consider linear functions and the combination of two linear functions: let \( k_1, k_2 \) be an interval in \( \mathbb{Q} \), \( L: [k_1, k_2] \to [0, 1] \) is defined as

\[
L_{[k_1, k_2]}(x; f_1, c, f_2) = \begin{cases} 
  f_1(x) & \text{if } k_1 \leq x \leq c \\
  f_2(x) & \text{if } c \leq x \leq k_2
\end{cases}
\]

where \( c \in [k_1, k_2] \), \( f_1 \) and \( f_2 \) are linear functions \( f_i: [k_1, k_2] \to [0, 1] \), \( f_i(x) = m_i x + q_i \), \( m_i, q_i \in \mathbb{Q} \), such that \( f_1(c) = f_2(c) \geq 0 \). Notice that for modifiers, we require that the domain is \([0, 1]\). Furthermore, note that the modifiers in [18] are a special case as additionally \( f_1(c) = f_2(c) \), \( m_1 > 0 \) and \( m_2 < 0 \) should hold. As an application of linear functions, we may define, e.g. the modifier very as \( L_{[0.1]}(x; \frac{2}{3} x, 0.75, 2x - 1) \). While the modifier \( m(x) = x^2 \) ([6]) cannot be bMIP-represented, the previous definition may be seen as an approximation of it. Multiple combinations of linear functions may be used to represent the membership function depicted in Figure 1.

For the sake of concrete illustration, we first show how to represent the combination of two linear functions as a bMIP. It will be then evident that any combination of more than two linear functions can be obtained in a similar way and, thus, the trapezoidal functions are just a special case. So, consider \( L_{[k_1, k_2]}(x; f_1, c, f_2) \). There are several cases to consider according to the value of \( m_i \) (\( < 0, 0 > 0 \)). In order to represent \( L \) as a bMIP, we have to define the graph \( g(L) = \{ (x_1, x_2): L(x_1) \geq x_2 \} \) as the solutions of a bMIP. However, as we may have negation on front of modifiers and fuzzy domain predicates, \( g(m) = \{ (x_1, x_2): L(x_1) \leq x_2 \} \) should be bMIP-representable as well. We just consider the former case as the latter can be developed in a similar way. We have that \( f_1(k_1) \geq 0 \) and \( f_2(k_2) \geq 0 \). Under this condition, \( g(L) \) can be split into two sets \( X_1 \) and \( X_2 \), \( g(L) = X_1 \cup X_2 \), where \( X_1 = \{ (x_1, x_2): f_1(x_1) \geq x_2, k_1 \geq x_1 \geq 0, \leq x_2 \leq 1 \} \), while \( X_2 = \{ (x_1, x_2): f_2(x_1) \geq x_2, c \leq x_1 \leq k_2, 0 \leq x_2 \leq 1 \} \). From the \( X_i \), we can build matrices \( A_i \) and rational positive vectors \( b^2_{1} \) (i, j = 1, 2) such that \( X_i \) can be written as the set \( X_i = \{ x: A_i x \geq b^2_{1} \} \). Now we introduce a 0-1 valued control variable \( y \) in order to merge the two sets \( X_1 \) and \( X_2 \) into a bMIP. Indeed, we define for vectors \( w^2_{1} \) of rational values \( X_{12} = \{ x: A_{12} x \geq (1 - y) \cdot b^2_{1} + y \cdot w^1_{1}, A_{12}^2 x \leq (1 - y) \cdot b^2_{1} + y \cdot w^2_{1}, A_{12}^3 x \geq y \cdot b^2_{1} + (1 - y) \cdot w^2_{2}, A_{12}^4 x \leq y \cdot b^2_{1} + (1 - y) \cdot w^2_{2} \} \). Then, it can be verified that there is a suitable choice of \( w^2_{2} \) such that for \( y = 0 \), \( X_{12} = X_1 \), while for \( y = 1 \) \( X_{12} = X_2 \) and, thus, \( X_{12} = g(L) \) from \( X_{12} \) a bMIP can easily be obtained. The graph \( g(L) \) can then be defined in a similar way. For instance, \( Y \), restricted to \([0, 30]\), can be defined as \( L_{[0,30]}(x; 1, 10, 30 - x/20) \) and, thus, it can be shown that \( g(L) \) is

\[
X_{12} = \{ (x_1, x_2): x_1 \leq 10(1 - y) + 30y, x_2 \geq (1 - y), x_1 \geq 10y, x_1 \leq 30y + 30(1 - y), x_1 + 20x_2 \geq 30y \}.
\]

This completes the first part. Now, in order to extend \( Y \) range over, say, \([0, 200]\) and not just over \([0, 30]\) (recall that \( Y \) \( x \) = 0 for \( x \geq 30 \)) we have to reapply the above procedure again to the sets \( X_{12} \) and \( X_3 \), where \( X_3 = \{ (x_1, x_2): x_1 \geq 30, x_2 = 0 \} \) (this will introduce another control variable \( y_1 \)), obtaining the set \( X_{123} \). Therefore, \( Y \) is bMIP representable with two control variables. In general, it can be verified that the above procedure can iteratively be applied to the union of \( n \geq 2 \) sets of the form \( X_i \), by means of the introduction of \( n - 1 \) control variables. In particular, trapezoidal functions can be represented as bMIP using at most four control variables (\( n = 5 \)).

The attentive reader will notice that a difficulty arises in representing crisp sets, such as e.g. \( \leq 18 \), as they present a discontinuity. To overcome partially to this situation, we may rely on a linear combination of the form \( L_{[18, +\epsilon]}(x; 1, 18, (18 + \epsilon - x)/\epsilon) \) for a sufficiently small \( \epsilon > 0 \) and then extend it to range over, say \([0, 150]\), by combining the previous function with \( f(x) = 0 \), for \( 18 + \epsilon \leq x \leq 150 \), in a similarly way as we did for \( Y \) (so, two control variables are needed).

B RULES FOR LUKASIEWICZ LOGIC

R.A. If \( (a, A, x) \in S_i \) and \( \alpha \) is an atomic assertion of the form \( a: A \) or \( a: b: R \) then \( S_{i+1} = S_i \cup \{ x: \alpha \geq 1 \} \).

R.A. If \( (a: \neg A, x) \in S_i \) then \( S_{i+1} = S_i \cup \{ x: \alpha \leq 1 \} \).

R.\sqcup. If \( (a: C \land D, x) \in S_i \) then \( S_{i+1} = S_i \cup \{ x: C, x \} \).

R.\sqcap. If \( (a: C \lor D, x) \in S_i \) then \( S_{i+1} = S_i \cup \{ x: D, x \} \).

R.\sqcup. If \( (a: C \sqcup D, x) \in S_i \) then \( S_{i+1} = S_i \cup \{ x: C, x \} \).

R.\sqcap. If \( (a: \exists R, C, x) \in S_i \) then \( S_{i+1} = S_i \cup \{ x: R, x \} \).
RV. If \( \{ \langle a \forall R.C, l_1 \rangle, \langle (a, b): R, l_2 \rangle \} \subseteq S_i \) then \( S_{i+1} = S_i \cup \{ \langle b: C, x \rangle, x \geq l_1 + l_2 - 1, x \leq y, l_1 + l_2 - 1 \leq y, l_1 + l_2 \geq y, x \in [0, 1], y \in \{0, 1\} \} \), where \( x \) is a new variable and \( y \) is a new control variable. The case for concrete roles is similar.

Rm. If \( \{ \langle a: m(C), l \rangle \in S_i \) then \( S_{i+1} = S_i \cup \gamma\langle a: C, l \rangle \), where the set \( \gamma\langle a: C, l \rangle \) is obtained from the bMIP representation (see appendix) of \( g(m) \) as follows: replace in \( g(m) \) all occurrences of \( x_2 \) with \( l \). Then resolve for \( x_1 \) and replace all occurrences of the form \( x_1 \geq l' \) with \( \langle a: l' \rangle \), while replace all occurrences the form \( x_1 \leq l' \) with \( \langle a: \mnf(C), 1-l' \rangle \).

Rm. The case \( \{ \langle a: \neg m(C), l \rangle \in \) is similar to rule Rm, where we use the bMIP representation of \( \bar{g}(m) \) in place of \( g(m) \).

Rd. If \( \{ \langle c: d, l \rangle \in S_i \) then \( S_{i+1} = S_i \cup \gamma\langle c: d, l \rangle \), where the set \( \gamma\langle c: d, l \rangle \) is obtained from the bMIP representation of \( g(d) \) by replacing all occurrences of \( x_2 \) with \( l \) and \( x_1 \) with \( x_c \).

Rd. The case \( \{ c: \neg d, l \} \in S_i \) is similar to rule Rd, where we use the bMIP representation of \( \bar{g}(d) \) in place of \( g(d) \).

Let us comment the RDF rule. By reasoning by case, for \( y = 0 \), we have \( x_1 \leq 1, x_1 + x_2 = l + 1 \), while for \( y = 1 \), we have \( l = 0, x_1 = 0 \). These two cases correspond to \( \max(0, x_1 + x_2 - 1) \geq l \), which is true if \( l = 0 \) or \( x_1 + x_2 - 1 \geq l \) \( y = 0 \) with \( x_1 + x_2 - 1 \geq 0 \). Therefore, the control variable \( y \) simulates the two alternatives of the max operator in the definition of conjunction. A similar argument applies to the other rules.

References


