Fuzzy Description Logic Programs

Umberto Straccia  
ISTI- CNR, Italy  
straccia@isti.cnr.it  

December 22, 2005

Abstract

Description Logic Programs (DLPs), which combine the expressive power of classical description logics and logic programs, are emerging as an important ontology description language paradigm. In this work, we present fuzzy DLPs, which extend DLPs by allowing the representation of vague/imprecise information.

Category: F.4.1: Mathematical Logic and Formal Languages: Mathematical Logic: [Logic and constraint programming]

Category: I.2.3: Artificial Intelligence: Deduction and Theorem Proving: [Logic programming]

Terms: Theory

Keywords: Fuzziness, Description Logics, Logic Programs

1 Introduction

Rule-based and object-oriented techniques are rapidly making their way into the infrastructure for representing and reasoning about the Semantic Web: combining these two paradigms emerges as an important objective.

Description Logic Programs (DLPs) [6, 7, 9, 10, 11, 20, 25], which combine the expressive power of classical Description Logics (DLs) and classical Logic Programs (LPs), are emerging as an important ontology description language paradigm. DLs capture the meaning of the most popular features of structured representation of knowledge, while LPs are powerful rule-based representation languages.

In this work, we present fuzzy DLPs, which is a novel extension of DLPs towards the representation of vague/imprecise information.

We proceed as follows. We first introduce the main notions related to fuzzy DLs and fuzzy LPs, and then show how both can be integrated, defining fuzzy DLPs in Section 3. Section 4 concludes and outlines future research.
2 Preliminaries

Fuzzy DLs. DLs [1] are a family of logics for representing structured knowledge. Each logic is identified by a name made of labels, which identify the operators allowed in that logic. Major DLs are the so-called logic $\mathcal{ALC}$ [34] (Attributive Language with Complement) and is used as a reference language whenever new concepts are introduced in DLs, $\mathcal{SHOIN}(\mathcal{D})$, which is the logic behind the ontology description language OWL DL and $\mathcal{SHIF}(\mathcal{D})$, which is the logic behind OWL LITE, a slightly less expressive language than OWL DL (see [18, 21]).

Fuzzy DLs [36, 40] extend classical DLs by allowing to deal with fuzzy/imprecise concepts, like “Calla is a very large, long white flower on thick stalks”, allowing to deal with so-called fuzzy or vague concepts, like “creamy”, “dark”, “hot”, “large” and “thick”, for which a clear and precise definition is not possible (another issue relates to the representation of terms like “very”, which are called fuzzy concepts modifiers). While in classical DLs concepts denotes sets, in fuzzy DLs fuzzy concepts denote fuzzy sets [44].

Syntax. While the method we rely on in combining fuzzy DLs with fuzzy LPs does not depend on the particular fuzzy DL of choice, to make the paper self-contained, we shall use here fuzzy $\mathcal{ALC}(\mathcal{D})$ [37], which is fuzzy $\mathcal{ALC}$ [36] extended with explicit represent membership functions for modifiers (such as “very”) and vague concepts (such as “Young”) [37]. We refer to [40] for fuzzy OWL DL and related work on fuzzy DLs.

Fuzzy $\mathcal{ALC}(\mathcal{D})$ allows explicitly to represent membership functions in the language via fuzzy concrete domains. 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 $d^0: \Delta_0^n \rightarrow [0, 1]$, which is a $n$-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 $[0, 1]_\mathbb{Q} = [0, 1] \cap \mathbb{Q}$. Concerning fuzzy predicates, there are many membership functions for fuzzy sets membership specification. However (see Figure 1), for $k_1 \leq a < b \leq c < d \leq k_2$ rational numbers, the trapezoidal $\text{trz}(a, b, c, d, [k_1, k_2])$, the triangular $\text{tri}(a, b, c, [k_1, k_2])$, the left-shoulder $\text{ls}(a, b, [k_1, k_2])$, the right-shoulder $\text{rs}(a, b, [k_1, k_2])$ and the crisp function $\text{cr}(a, b, [k_1, k_2])$ are simple, yet most frequently used to specify membership degrees and are those we are considering in this paper. To simplify the notation, we may omit the domain range, and write, e.g. $\text{cr}(a, b)$ in place of $\text{cr}(a, b, [k_1, k_2])$, whenever the domain range is not important. For instance, the concept “less than 18 year old” can be defined as a crisp concept $\text{cr}(0, 18)$, while Young, denoting the degree of youngness of a person’s age, may be defined as Young $= \text{ls}(10, 30)$. We also consider fuzzy modifiers in fuzzy $\mathcal{ALC}(\mathcal{D})$. Fuzzy modifiers, like very, more or less and slightly, apply to fuzzy sets to change their membership function. Formally, a modifier is a function $f_m: [0, 1] \rightarrow [0, 1]$.

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 constant names (denoted $a$), concrete constant names (denoted $c$) and modifiers (denoted $m$). $R_a$ contains a non-empty subset
Figure 1: (a) Trapezoidal; (b) Triangular; (c) Left-shoulder; (d) Right-shoulder; (e) Crisp

\[ C \rightarrow \top \mid \bot \mid A \mid C_1 \cap C_2 \mid C_1 \cup C_2 \mid \neg C \mid \forall R.C \mid \exists R.C \mid \forall T.D \mid \exists T.D \mid m(C) \]
\[ D \rightarrow d \mid \neg d \]
\[ m \rightarrow \ln(a, b, c, d) \]
\[ d \rightarrow \text{trz}(a, b, c, d, [k_1, k_2]) \mid \text{tri}(a, b, c, [k_1, k_2]) \mid \ln(a, b, [k_1, k_2]) \mid \text{rs}(a, b, [k_1, k_2]) \mid \text{cr}(a, b, [k_1, k_2]) \]

Figure 2: \( ALC(D) \) concepts

\( F_a \) 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 \( ALC(D) \) concepts is defined by the syntactic rules \( d \) is a unary fuzzy predicate) in Figure 2. A TBox \( T \) consists of a finite set of terminological axioms of the form \( C_1 \sqsubseteq C_2 \) (\( C_1 \) is sub-concept of \( C_2 \)) or \( A = C \) (\( A \) is defined as the concept \( C \)), where \( A \) is a concept name and \( C \) is concept. Using axioms we may define the concepts of a minor and young person as

\[
\text{Minor} = \text{Person} \sqcap \exists \text{age} \leq 18
\]

\[
\text{YoungPerson} = \text{Person} \sqcap \exists \text{age} \text{Young}
\]

We also allow to formulate statements about constants. A concept-, role-assertion axiom and an constant (in)equality axiom has the form \( a : C \) (a is an instance of \( C \)), \( (a, b) : R \) (a is related to \( b \) via \( R \)), \( a \approx b \) (\( a \) and \( b \) are equal) and \( a \not\approx b \), respectively, where \( a, b \) are abstract constants. For \( n \in [0, 1]_Q \), an ABox \( A \) consists of a finite set of constant (in)equality axioms, and fuzzy concept and fuzzy role assertion axioms of the form \( \langle \alpha, n \rangle \), where \( \alpha \) is a concept or role assertion. Informally, \( \langle \alpha, n \rangle \) constrains the truth degree of \( \alpha \) to be greater or equal to \( n \). A fuzzy \( ALC(D) \) knowledge base \( \mathcal{K} = \langle T, A \rangle \) consists of a TBox \( T \) and an ABox \( A \).

\textbf{Semantics.} We recall here the main notions related to fuzzy DLs (for more on fuzzy DLs, see [36, 40]). The main idea is that an assertion \( a : C \), rather being interpreted as either true or false, will be mapped into a truth value \( c \in [0, 1]_Q \). The intended meaning is that \( c \) indicates to which extend ‘\( a \) is a \( C \)’. Similarly for role names. Formally, a fuzzy interpretation \( \mathcal{I} \) with respect to a concrete domain \( D \) is a pair \( \mathcal{I} = (\Delta^C, \Delta^R) \) consisting of a non empty set \( \Delta^C \) (called the domain), disjoint from \( \Delta_D \), and of a fuzzy interpretation function \( \mathcal{I}^C \) that assigns (i) to each abstract concept \( C \in C \) a function \( C^\mathcal{I} : \Delta^C \rightarrow [0, 1] \); (ii) to each abstract role \( R \in R_a \) a function \( R^\mathcal{I} : \Delta^R \times \Delta^C \rightarrow [0, 1] \); (iii) to each abstract
Regarding terminological axioms, we also introduce degrees of subsumption. I

<table>
<thead>
<tr>
<th></th>
<th>Lukasiewicz Logic</th>
<th>Gödel Logic</th>
<th>Product Logic</th>
<th>Zadeh semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="https://via.placeholder.com/150" alt="image" /> (x)</td>
<td>(1 - x)</td>
<td>(d(x)) if (x = 0) then 1 else 0</td>
<td>(d(x)) if (x \geq 0) then 1 else 0</td>
<td>(1 - x)</td>
</tr>
<tr>
<td>(x \land y)</td>
<td>(\max(x + y - 1, 0))</td>
<td>(\min(x, y))</td>
<td>(x \cdot y)</td>
<td>(\min(x, y))</td>
</tr>
<tr>
<td>(x \lor y)</td>
<td>(\min(x + y, 1))</td>
<td>(\max(x, y))</td>
<td>(x + y - x \cdot y)</td>
<td>(\max(x, y))</td>
</tr>
<tr>
<td>(x \Rightarrow y)</td>
<td>if (x \leq y) then 1 else (1 - x + y)</td>
<td>if (x \leq y) then 1 else (y/x)</td>
<td>if (x \leq y) then 1 else (y/x)</td>
<td>(\max(1 - x, y))</td>
</tr>
</tbody>
</table>

Table 1: Typical connective interpretation.

feature \(r \in F_{\alpha}\) a partial function \(r^I: \Delta^I \times \Delta^I \rightarrow [0, 1]\) such that for all \(u \in \Delta^I\) there is an unique \(w \in \Delta^I\) on which \(r^I(u, w)\) is defined; (iv) to each abstract constant \(a \in I_a\) an element in \(\Delta^I\); (v) to each concrete constant \(c \in I_c\) an element in \(\Delta^0\); (vi) to each concrete role \(T \in R\), a function \(T^I: \Delta^I \times \Delta^0 \rightarrow [0, 1]\); (vii) to each abstract role \(t \in F_t\), a partial function \(t^I: \Delta^I \times \Delta^0 \rightarrow [0, 1]\) such that for all \(u \in \Delta^I\) there is an unique \(o \in \Delta^0\) on which \(t^I(u, o)\) is defined; (viii) to each modifier \(m \in M\) the function \(f_m: [0, 1] \rightarrow [0, 1]\); (ix) to each unary concrete predicate \(d\) the fuzzy relation \(d^I: \Delta^0 \rightarrow [0, 1]\) and to \(\neg\) the negation of \(d^I\). To extend the interpretation function to complex concepts, we use so-called t-norms (interpreting conjunction), s-norms (interpreting disjunction), negation function (interpreting negation), and implication function (interpreting implication) [17]. In Table 1 we report most used combinations of norms.

The mapping \(I\) is then extended to concepts and roles as follows (where \(u \in \Delta^I\): \(\top^I(u) = 1\), \(\bot^I(u) = 0\),

\[
\begin{align*}
(C_1 \cap C_2)^I(u) &= C_1^I(u) \land C_2^I(u) \\
(C_1 \cup C_2)^I(u) &= C_1^I(u) \lor C_2^I(u) \\
(\neg C)^I(u) &= \neg C^I(u) \\
(mC)^I(u) &= f_m(C^I(u)) \\
(\forall R.C)^I(u) &= \inf_{w \in \Delta^I} R^I(u, w) \Rightarrow C^I(u) \\
(\exists R.C)^I(u) &= \sup_{w \in \Delta^I} R^I(u, w) \land C^I(u) \\
(\forall T.D)^I(u) &= \inf_{o \in \Delta^0} T^I(u, o) \Rightarrow D^I(o) \\
(\exists T.D)^I(u) &= \sup_{o \in \Delta^0} T^I(u, o) \land D^I(o).
\end{align*}
\]

The mapping \(I\) is extended to assertion axioms as follows (where \(a, b \in I_a\):

\[(a: C)^I = C^I(a, b)\text{ and }((a: b): R)^I = R^I(a, b, I^I).\]

The notion of satisfiability of a fuzzy axiom \(E\) by a fuzzy interpretation \(I\), denoted \(I \models E\), is defined as follows: \(I \models C_1 \subseteq C_2\) if for all \(u \in \Delta^I\), \(C_1^I(u) \leq C_2^I(u)\); \(I \models A = C\) if for all \(u \in \Delta^I\), \(A^I(u) = C^I(u)\); \(I \models \langle a, n \rangle\) if \(a^n \geq n\); \(I \models a \approx b\) if \(a^I \approx b^I\); and \(I \models a \not\approx b\) if \(a^I \not\approx b^I\). The notion of satisfiability (is model) of a knowledge base \(K = \langle T, A\rangle\) and entailment of an assertional axiom is straightforward.

Concerning terminological axioms, we also introduce degrees of subsumption. We say that \(K\) entails \(C_1 \subseteq C_2\) to degree \(n \in [0, 1]\), denoted \(K \models C_1 \subseteq C_2, n\) if for every model \(T\) of \(K\), \([\inf_{u \in \Delta^I} C_1^I(u) \Rightarrow C_2^I(u)] \geq n\).

**Example 1** (40) Consider the following simplified excerpt of a knowledge base about cars:

\[
\begin{align*}
\text{SportsCar} &= \exists \text{speed.very}(High), \\
\text{mg_mgb} &= \exists \text{speed} \leq 170, 1, \\
\text{ferrari_enzo} &= \exists \text{speed} > 350, 1, \\
\text{audi_tt} &= \exists \text{speed} \approx 243, 1
\end{align*}
\]
speed is a concrete feature. The fuzzy domain predicate High has membership function $\text{High} = rs(80, 250)$. It can be shown that

$$K \models (\text{mg_mgb}: \neg\text{SportsCar}, 0.72)$$

$$K \models (\text{ferrari_enzo: SportsCar}, 1)$$

$$K \models (\text{audi_tt: SportsCar}, 0.92)$$

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 $K \models (\text{Minor} \sqsubseteq \text{YoungPerson}, 0.5)$ holds.

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 $\text{glb}(K, \alpha)$, is $\text{glb}(K, \alpha) = \sup\{n: K \models (\alpha, n)\}$, where $\sup \emptyset = 0$. Determining the $\text{glb}$ 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 $\text{glb}(K, \alpha) \geq n$. Therefore, the BDB problem is the major problem we have to consider in fuzzy $\mathcal{ALC}(D)$.

**Fuzzy LPs.** The management of imprecision in logic programming has attracted the attention of many researchers and numerous frameworks have been proposed. Essentially, they differ in the underlying truth space (e.g. Fuzzy set theory [2, 8, 22, 35, 42, 43], Multi-valued logic [3, 4, 5, 13, 14, 23, 24, 26, 27, 28, 29, 30, 31, 33, 32, 39, 38]), and how imprecision values, associated to rules and facts, are managed.

**Syntax.** We consider here a very general form of the rules [38, 39]:

$$A \leftarrow f(B_1, ..., B_n),$$  \hspace{1cm} (3)

where $f \in \mathcal{F}$ is an $n$-ary computable monotone function $f: [0, 1]^n \rightarrow [0, 1]$, and $B_i$ are atoms. Each rule may have a different $f$. An example of rule is

$$s \leftarrow \min(p, q) \cdot \max(\neg r, 0.7) + v,$$

where $p, q, r, s$ and $v$ are atoms. Computationally, given an assignment $I$ of values to the $B_i$, the value of $A$ is computed by stating that $A$ is at least as true as $f(I(B_1), ..., I(B_n))$. The form of the rules is sufficiently expressive to encompass all approaches to fuzzy logic programming we are aware of. We assume that the standard functions $\wedge$ (meet) and $\vee$ (join) belong to $\mathcal{F}$. Notably, $\wedge$ and $\vee$ are both monotone. We call $f \in \mathcal{F}$ a truth combination function, or simply combination function $^1$. We recall that an atom, denoted $A$, is an expression of the form $P(t_1, \ldots, t_n)$, where $P$ is an $n$-ary predicate symbol and all $t_i$ are terms, i.e. a constant or a variable. A generalized normal logic program, or simply normal logic program, denoted with $\mathcal{P}$, is a finite set of rules. The Herbrand universe $H_{\mathcal{P}}$ of $\mathcal{P}$ is the set of constants appearing in $\mathcal{P}$. If there is no constant symbol in $\mathcal{P}$ then consider $H_{\mathcal{P}} = \{a\}$, where $a$ is an arbitrary chosen constant. The Herbrand base $B_{\mathcal{P}}$ of $\mathcal{P}$ is the set of ground

---

$^1$Due to lack of space, we do not deal with non-monotonic negation here, though we can managed is as in [38].
instantiations of atoms appearing in \( P \) (ground instantiations are obtained by replacing all variable symbols with constants of the Herbrand universe). Given \( P \), the generalized normal logic program \( P^* \) is constructed as follows: (i) set \( P^* \) to the set of all ground instantiations of rules in \( P \); (ii) if an atom \( A \) is not head of any rule in \( P^* \), then add the rule \( A \leftarrow 0 \) to \( P^* \) (it is a standard practice in logic programming to consider such atoms as \textit{false}); (iii) replace several rules in \( P^* \) having same head, \( A \leftarrow \varphi_1, A \leftarrow \varphi_2, \ldots \) with \( A \leftarrow \varphi_1 \lor \varphi_2 \lor \ldots \) (recall that \( \lor \) is the join operator of the truth lattice in infix notation). Note that in \( P^* \), each atom appears in the head of \textit{exactly one} rule.

**Semantics.** An interpretation \( I \) of a logic program is a mapping from atoms to members of \([0,1]_Q\). \( I \) is extended from atoms to the interpretation of rule bodies as follows: \( I(f(B_1, \ldots, B_n)) = f(I(B_1), \ldots, I(B_n)) \). The ordering \( \leq \) is extended from \([0,1]_Q\) to the set of all interpretations point-wise: (i) \( I_1 \leq I_2 \) iff \( I_1(A) \leq I_2(A) \), for every ground atom \( A \). With \( I_\bot \) we denote the bottom interpretation under \( \leq \) (it maps any atom into 0).

An interpretation \( I \) is a model of a logic program \( P \), denoted by \( I \models P \), iff for all \( A \leftarrow \varphi \in P^* \), \( I(\varphi) \leq I(A) \) holds. The semantics of a logic program \( P \) is determined by the least model of \( P \), \( M_P = \min\{I : I \models P\} \). The existence and uniqueness of \( M_P \) is guaranteed by the fixed-point characterization, by means of the immediate consequence operator \( \Phi_P \). For an interpretation \( I \), for any ground atom \( A \), \( \Phi_P(I)(A) = I(\varphi) \), where \( A \leftarrow \varphi \in P^* \). We can show that the function \( \Phi_P \) is monotone, the set of fixed-points of \( \Phi_P \) is a complete lattice and, thus, \( \Phi_P \) has a least fixed-point and \( I \) is a model of a program \( P \) iff \( I \) is a fixed-point of \( \Phi_P \). Therefore, the minimal model of \( P \) coincides with the least fixed-point of \( \Phi_P \), which can be computed in the usual way by iterating \( \Phi_P \) over \( I_\bot \) [38, 39].

**Example 3 ([43])** In [43], Fuzzy Logic Programming is proposed, where rules have the form \( A \leftarrow f(A_1, \ldots, A_n) \) for some specific \( f \). [43] is just a special case of our framework. As an illustrative example consider the following scenario. Assume that we have the following facts, represented in the tables below. There are hotels and conferences, their locations and the distance among locations.

<table>
<thead>
<tr>
<th>HasLocationH</th>
<th>HotelID</th>
<th>HasLocationH</th>
<th>HotelID</th>
</tr>
</thead>
<tbody>
<tr>
<td>h1</td>
<td>h1</td>
<td>h1</td>
<td>h1</td>
</tr>
<tr>
<td>h2</td>
<td>h2</td>
<td>h2</td>
<td>h2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HasLocationC</th>
<th>ConferenceID</th>
<th>HasLocationC</th>
<th>ConferenceID</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>c1</td>
<td>c1</td>
<td>c1</td>
</tr>
<tr>
<td>c2</td>
<td>c2</td>
<td>c2</td>
<td>c2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Distance</th>
<th>HasLocationH</th>
<th>HasLocationH</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>h11</td>
<td>c1</td>
<td>h11</td>
<td>300</td>
</tr>
<tr>
<td>h12</td>
<td>c1</td>
<td>h12</td>
<td>500</td>
</tr>
<tr>
<td>h11</td>
<td>c11</td>
<td>h11</td>
<td>500</td>
</tr>
<tr>
<td>h12</td>
<td>c12</td>
<td>h12</td>
<td>700</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

6
Now, suppose that our query is to find hotels close to the conference venue, labeled $c_1$. We may formulate our query as the rule:

$$\text{Query}(h) \leftarrow \min(\text{HasLocationH}(h, hl), \text{HasLocationC}(c_1, cl), \text{Distance}(hl, cl, d), \text{Close}(d))$$

where $\text{Close}(x)$ is defined as $\text{Close}(x) = \max(0, 1 - x/1000)$. As a result to that query we get a ranked list of hotels as shown in the table below.

<table>
<thead>
<tr>
<th>HotelID</th>
<th>Closeness degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>h1</td>
<td>0.7</td>
</tr>
<tr>
<td>h2</td>
<td>0.25</td>
</tr>
</tbody>
</table>

### 3 Fuzzy DLPs

In this section we introduce fuzzy Description Logic Programs (fuzzy DLPs), which are a combination of fuzzy DLs with fuzzy LPs. In the classical semantics setting, there are mainly three approaches (see, [12, 15], for an overview), the so-called axiom-based approach (e.g., [20, 25]) and the DL-log approach (e.g., [7, 9, 10]) and the autoepistemic approach (e.g., [6, 11]). We are not going to discuss in this section these approaches. The interested reader may see [41]. We just point out that in this paper we follow the DL-log approach, in which rules may not modify the extension of concepts and DL atoms and roles appearing the body of a rule act as procedural calls to the DL component.

We would like to note that the unique combination of DL and LPs for the management of imprecision we are aware of is [41]. The major problems behind [41] rely on the computational part. Indeed, [41] requires that the so-called annotation terms (see [23]) are grounded, which makes the approach hardly feasible in practice. We do not have here such restrictions.

**Syntax.** We assume that the description logic component and the rules component share the same alphabet of constants. Rules are as for fuzzy LPs except that now atoms and roles may appear in the rule body. We assume that no rule head atom belongs to the DL signature. For ease the readability, in case of ambiguity, DL predicates will have a DL superscript in the rules. Note that in [9] a concept inclusion may appear in the body of the rule. We will not deal with this feature. A fuzzy Description Logic Program (fuzzy DLP) is a tuple $\mathcal{DP} = \langle \mathcal{K}, \mathcal{P} \rangle$, where $\mathcal{K}$ is a fuzzy DL knowledge base and $\mathcal{P}$ is a fuzzy logic program. For instance, the following is a fuzzy DLP:

```plaintext
LowCarPrize(x) ← \min(\text{made\_by}(x, y), \text{ChineseCarCompany}\_DL(y)), \text{has\_prize}(x, z), \text{LowPrize}\_DL(z)
made\_by(x, y) ← \text{makes}\_DL(y, x),
LowPrize = 1s(5.000, 15.000)
ChineseCarCompany = (\exists\text{has\_location}\_China), \text{makes}\_Car
```
with meaning: a chinese car company is located in china, makes cars, which are sold as low prize cars. Low prize is defined as a fuzzy concept with left-shoulder membership function.

**Semantics.** We recall that in the DL-log approach, a DL atom appearing in a rule body acts as a query to the underlying DL knowledge base (see [9]). So, consider a fuzzy DLP \( \mathcal{DP} = (\mathcal{K}, \mathcal{P}) \). The Herbrand universe of \( \mathcal{P} \), denoted \( H_\mathcal{P} \) is the set of constants appearing in \( \mathcal{DP} \) (if no such constant symbol exists, \( H_\mathcal{P} = \{c\} \) for an arbitrary constant symbol c from the alphabet of constants).

The Herbrand base of \( \mathcal{P} \), denoted \( B_\mathcal{P} \), is the set of all ground atoms built up from the non-DL predicates and the Herbrand universe of \( \mathcal{P} \). Then, the definition of \( \mathcal{P}^* \) is as for fuzzy LPs. An interpretation \( I \) w.r.t. \( \mathcal{DP} = (\mathcal{K}, \mathcal{P}) \) is a function \( I : B_\mathcal{P} \to [0, 1]_Q \) mapping non-DL atoms into \([0, 1]_Q\). We say that \( I \) is a model of a \( \mathcal{DP} = (\mathcal{K}, \mathcal{P}) \) iff \( I^K \models \mathcal{P} \), where

1. \( I^K \models \mathcal{P} \) iff for all \( A \leftarrow \varphi \in \mathcal{P}^* \), \( I^K(\varphi) \leq I^K(A) \);
2. \( I^K(f(A_1, \ldots, A_n)) = f(I^K(A_1), \ldots, I^K(A_n)) \);
3. \( I^K(P(t_1, \ldots, t_n)) = I(P(t_1, \ldots, t_n)) \) for all ground non-DL atoms \( P(t_1, \ldots, t_n) \);
4. \( I^K(A(a)) = \text{glb}(\mathcal{K}, a : A) \) for all ground DL atoms \( A(a) \);
5. \( I^K(R(a, b)) = \text{glb}(\mathcal{K}, (a, b) : R) \) for all ground DL roles \( R(a, b) \).

Note how in Points 4. and 5. the interpretation of a DL-atom and role depends on the DL-component only. Finally, we say that \( \mathcal{DP} = (\mathcal{K}, \mathcal{P}) \) entails a ground atom \( A \), denoted \( \mathcal{DP} \models A \), iff \( I \models A \) whenever \( I \models \mathcal{DP} \).

For instance, assume that together with the \( \mathcal{DP} \) about low prize cars we have the following instances, where \( l1 \) and \( l2 \) are located in China and \( c1 \) and \( c2 \) are cars.

<table>
<thead>
<tr>
<th>CarCompany</th>
<th>has_location</th>
<th>Makes</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>l1</td>
<td>c1</td>
</tr>
<tr>
<td>c2</td>
<td>l2</td>
<td>c2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

If the prizes are as in the left table below then the degree of the car prizes is depicted in the right table below. Note that due to the definition of chinese car companies, \( c1 \) and \( c2 \) are chinese car companies.

<table>
<thead>
<tr>
<th>Prize</th>
<th>LowPriceCar</th>
</tr>
</thead>
<tbody>
<tr>
<td>car1</td>
<td>10,000</td>
</tr>
<tr>
<td>car2</td>
<td>7,000</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Interestingly, it is possible to adapt the standard results of Datalog to our case, which say that a satisfiable description logic program \( \mathcal{DP} \) has a minimal model \( M_{\mathcal{DP}} \) and entailment can be reduced to model checking in this minimal model.
Proposition 1 Let $\mathcal{D}P = \langle \mathcal{K}, \mathcal{P} \rangle$ be a fuzzy DLP. If $\mathcal{D}P$ is satisfiable, then there exists a unique model $M_{\mathcal{D}P}$ such that $M_{\mathcal{D}P} \leq I$ for all models $I$ of $\mathcal{D}P$. Furthermore, for any ground atom $A$, $\mathcal{D}P \models A$ iff $M_{\mathcal{D}P} \models A$.

The minimal model can be computed as the least fixed-point of the following monotone operator. Let $\mathcal{D}P = \langle \mathcal{K}, \mathcal{P} \rangle$ be a fuzzy DLP. Define the operator $T_{\mathcal{D}P}$ on interpretations as follows: for every interpretation $I$, for all ground atoms $A \in \mathcal{B}_{\mathcal{P}}$, given $A \leftarrow \varphi \in \mathcal{P}^*$

$$T_{\mathcal{D}P}(I)(A) = \ell^\mathcal{K}(\varphi) .$$

Then it can easily be shown that $T_{\mathcal{D}P}$ is monotone, i.e. $I \leq I'$ implies $T_{\mathcal{D}P}(I) \leq T_{\mathcal{D}P}(I')$, and, thus, by the Knaster-Tarski Theorem $T_{\mathcal{D}P}$ has a least fixed-point, which can be computed as a fixed-point iteration of $T_{\mathcal{D}P}$ starting with $I_\bot$.

Reasoning. From a reasoning point of view, to solve the entailment problem we proceed as follows. Given $\mathcal{D}P = \langle \mathcal{K}, \mathcal{P} \rangle$, we first compute for all DL atoms $A(a)$ occurring in $\mathcal{P}^*$, the greatest truth lower bound, i.e. $n_{A(a)} = \text{glb(} \mathcal{K}, a : A \text{)}$. Then we add the rule $A(a) \leftarrow n_{A(a)}$ to $\mathcal{P}$, establishing that the truth degree of $A(a)$ is at least $n_{A(a)}$ (similarly for roles). Finally, we can rely on a theorem prover for fuzzy LPs only either using a usual bottom-up computation or a top-down computation for logic programs [3, 38, 39, 43]. Of course, one has to be sure that both computations, for the fuzzy DL component and for the fuzzy LP component, are supported. With respect to the logic presented in this paper, we need the reasoning algorithm described in [37] for fuzzy DLs component \footnote{However, sub-concept specification in terminological axioms are of the form $A \sqsubseteq C$ only, where $A$ is a concept name and neither cyclic definitions are allowed nor may there be more than one definition per concept name $A$.}, while we have to use [38, 39] for the fuzzy LP component.

We conclude by mentioning that by relying on [38], the whole framework extends to fuzzy description normal logic programs as well (non-monotone negation is allowed in the logic programming component).

4 Conclusions

We integrated the management of imprecision into a highly expressive family of representation languages, called fuzzy Description Logic Programs, resulting from the combination of fuzzy Description Logics and fuzzy Logic Programs. We defined syntax, semantics, declarative and fixed-point semantics of fuzzy DLPs. We also detailed how query answering can be performed by relying on the combination of currently known algorithms, without any significant additional effort.

Our motivation is inspired by its application in the Semantic Web, in which both aspects of structured and rule-based representation of knowledge are becoming of interest [16, 19].

There are some appealing research directions. At first, it would certainly be of interest to investigate about reasoning algorithm for fuzzy description
logic programs under the so-called axiomatic approach. Currently, very few is known about that. Secondly, while there is a huge literature about fuzzy logic programming and many-valued programming in general, very little is known in comparison about fuzzy DLs. This area may deserve more attention.

References


