Fuzzy Ontology Representation using OWL 2 ☆

Fernando Bobillo[a], Umberto Straccia[b]

[a] *Department of Computer Science and Systems Engineering, University of Zaragoza, Spain*
[b] *Istituto di Scienza e Tecnologie dell'Informazione (ISTI - CNR), Pisa, Italy*

Abstract

The need to deal with vague information in Semantic Web languages is rising in importance and, thus, calls for a standard way to represent such information. We may address this issue by either extending current Semantic Web languages to cope with vagueness, or by providing a procedure to represent such information within current standard languages and tools. In this work, we follow the latter approach, by identifying the syntactic differences that a fuzzy ontology language has to cope with, and by proposing a concrete methodology to represent fuzzy ontologies using OWL 2 annotation properties. We also report on the prototypical implementations.

Key words: Fuzzy OWL 2, Fuzzy Ontologies, Fuzzy Languages for the Semantic Web, Fuzzy Description Logics

1. Introduction

Today, there is a growing interest in the development of knowledge representation formalisms able to deal with uncertainty, which is a very common requirement in real world applications. Despite the undisputed success of ontologies, classical ontology languages are not appropriate to deal with vagueness or imprecision in the knowledge, which is inherent to most of the real world application domains [29].

Since fuzzy set theory and fuzzy logic [30] are suitable formalisms to handle these types of knowledge, fuzzy ontologies emerge as useful in several applications, ranging from (multimedia) information retrieval to image interpretation, ontology mapping, matchmaking, decision making, or the Semantic Web [19].

Description Logics (DLs for short) [1] are a family of logics for representing structured knowledge. Each logic is denoted by using a string of capital letters which identify the constructors of the logic and therefore its complexity. DLs

☆This paper is a revised and considerably extended version of "Representing Fuzzy Ontologies in OWL 2", published in the Proceedings of the 19th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2010).

Email addresses: fbobillo@unizar.es (Fernando Bobillo), straccia@isti.cnr.it (Umberto Straccia)

have proved to be very useful as ontology languages. For instance, the language OWL 2, which has very recently become a W3C Recommendation for ontology representation [9, 17], is equivalent to the DL $\mathcal{SROIQ}(\mathbf{D})$.

Several fuzzy extensions of DLs can be found in the literature (see the survey in [15]) and some fuzzy DL reasoners have been implemented, such as fuzzyDL [5], DeLorean [2] and Fire [20]. Not surprisingly, each reasoner uses its own fuzzy DL language for representing fuzzy ontologies and, thus, there is a need for a standard way to represent such information.

A first possibility would be to adopt as an standard one of the fuzzy extensions of the languages OWL and OWL 2 that have been proposed [10, 21, 22]. However, we do not expect a fuzzy OWL extension to become a W3C proposed standard in the near future. Furthermore, we argue that current fuzzy extensions are not expressive enough, as they only provide syntactic modifications in the ABox.

In this work, we propose to use OWL 2 itself to represent fuzzy ontologies. More precisely, we use OWL 2 annotation properties to encode fuzzy $\mathcal{SROIQ}(\mathbf{D})$ ontologies. The use of annotation properties makes possible *(i)* to use current OWL 2 editors for fuzzy ontology representation, and *(ii)* that OWL 2 reasoners discard the fuzzy part of a fuzzy ontology, producing the same results as if would not exist. Additionally, we identify the syntactic differences that a fuzzy ontology language has to cope with, and show how to address them using OWL 2 annotation properties.

The remainder of this paper is organized as follows. In Section 3 we present a fuzzy extension of DL $\mathcal{SROIQ}(\mathbf{D})$, the logic behind OWL 2, including some additional constructs, peculiar to fuzzy logic. Section 4 discusses how to encode it using OWL 2 language. Section 5 illustrates the methodology with some application problems. Section 6 discusses the implementation status of our approach and compares it with the related work. Finally, Section 7 sets out some conclusions and ideas for future research.

2. Fuzzy Logic

Fuzzy set theory and fuzzy logic were proposed by L. Zadeh [30] to manage imprecise and vague knowledge. While in classical set theory elements either belong to a set or not, in fuzzy set theory elements can belong to a set to some degree. More formally, let X be a set of elements called the reference set. A *fuzzy subset* A of X is defined by a membership function $\mu_A(x)$, or simply $A(x)$, which assigns any $x \in X$ to a value in the interval of real numbers between 0 and 1. As in the classical case, 0 means no-membership and 1 full membership, but now a value between 0 and 1 represents the extent to which x can be considered as an element of X.

Changing the usual true/false convention leads to a new concept of statement, whose compatibility with a given state of facts is a matter of degree, usually called the *degree of truth* of the statement. In this article we will consider *fuzzy statements* of the form $\phi \geq \alpha$ or $\phi \leq \beta$, where $\alpha, \beta \in [0, 1]$ [11] and ϕ

is a statement. This encodes the fact that the degree of truth of ϕ is *at least* l (resp. *at most* u). For example, $\mathsf{ripeTomato} \geq 0.9$ says that we have a rather ripe tomato (the degree of truth of $\mathsf{ripeTomato}$ is at least 0.9).

All crisp set operations are extended to fuzzy sets. The intersection, union, complement and implication set operations are performed by a t-norm function, a t-conorm function, a negation function and an implication function, respectively. These operations can be grouped in families or fuzzy logics. It is well known that different fuzzy logics have different properties [11].

There are three main fuzzy logics: Łukasiewicz, Gödel, and Product. The importance of these three fuzzy logics is due the fact that any continuous t-norm can be obtained as a combination of Łukasiewicz, Gödel, and Product t-norm [16]. It is also common to consider the fuzzy connectives originally considered by Zadeh (Gödel conjunction and disjunction, Łukasiewicz negation and Kleene-Dienes implication), which is sometimes known as Zadeh fuzzy logic. Table 1 shows these four fuzzy logics: Zadeh, Łukasiewicz, Gödel, and Product.

Table 1: Some popular fuzzy logics

Family	t-norm $\alpha \otimes \beta$	t-conorm $\alpha \oplus \beta$	negation $\ominus\alpha$	implication $\alpha \Rightarrow \beta$
Zadeh	$\min\{\alpha, \beta\}$	$\max\{\alpha, \beta\}$	$1 - \alpha$	$\max\{1 - \alpha, \beta\}$
Gödel	$\min\{\alpha, \beta\}$	$\max\{\alpha, \beta\}$	$\begin{cases} 1, & \alpha = 0 \\ 0, & \alpha > 0 \end{cases}$	$\begin{cases} 1 & \alpha \leq \beta \\ \beta, & \alpha > \beta \end{cases}$
Łukasiewicz	$\max\{\alpha + \beta - 1, 0\}$	$\min\{\alpha + \beta, 1\}$	$1 - \alpha$	$\min\{1 - \alpha + \beta, 1\}$
Product	$\alpha \cdot \beta$	$\alpha + \beta - \alpha \cdot \beta$	$\begin{cases} 1, & \alpha = 0 \\ 0, & \alpha > 0 \end{cases}$	$\begin{cases} 1 & \alpha \leq \beta \\ \beta/\alpha, & \alpha > \beta \end{cases}$

A fuzzy set C is *included* in another fuzzy set D iff $\forall x \in X, \mu_C(x) \leq \mu_D(x)$. According to this definition, which is usually called Zadeh's set inclusion, fuzzy set inclusion is a yes-no question. In order to overcome this, other definitions have been proposed. For example, the degree of inclusion of C in D can be computed using some implication function as $\inf_{x \in X} \mu_C(x) \Rightarrow \mu_D(x)$. Note that these two approaches are equivalent under Rescher implication, defined as $\alpha \Rightarrow \beta = 1$ iff $\alpha \leq \beta$, or $\alpha \Rightarrow \beta = 0$ otherwise.

A (binary) *fuzzy relation* R over two countable classical sets X and Y is a function $R\colon X \times Y \to [0, 1]$. The *inverse* of R is the function $R^{-1}\colon Y \times X \to [0, 1]$ with membership function $R^{-1}(y, x) = R(x, y)$, for every $x \in X$ and $y \in Y$. The *composition* of two fuzzy relations $R_1\colon X \times Y \to [0, 1]$ and $R_2\colon Y \times Z \to [0, 1]$ is defined as $(R_1 \circ R_2)(x, z) = \sup_{y \in Y} R_1(x, y) \otimes R_2(y, z)$. A fuzzy relation R is *transitive* iff $R(x, z) \geq (R \circ R)(x, z)$.

A fuzzy interpretation $\mathcal{I}$ *satisfies* a fuzzy statement $\phi \geq l$ (resp., $\phi \leq u$) or $\mathcal{I}$ is a *model* of $\phi \geq l$ (resp., $\phi \leq u$), denoted $\mathcal{I} \models \phi \geq l$ (resp., $\mathcal{I} \models \phi \leq u$), iff $\mathcal{I}(\phi) \geq l$ (resp., $\mathcal{I}(\phi) \leq u$). The notions of satisfiability and logical consequence are defined in the standard way. We say that $\phi \geq l$ is a *tight logical consequence* of a set of fuzzy statements $\mathcal{K}$ iff l is the infimum of $\mathcal{I}(\phi)$ subject to all models $\mathcal{I}$ of $\mathcal{K}$. Notice that the latter is equivalent to $l = \sup\{r \mid \mathcal{K} \models \phi \geq r\}$. For reasoning algorithms for fuzzy propositional and First-Order Logics see [11].

3. The Fuzzy DL $\mathcal{SROIQ}(\mathbf{D})$

In this section we describe the fuzzy DL $\mathcal{SROIQ}(\mathbf{D})$, a subset of the language presented in [8], which was inspired by the logics presented in [4, 5, 26]. Here, concepts denote fuzzy sets of individuals and roles denote fuzzy binary relations. Axioms are also extended to the fuzzy case and some of them hold to a degree.

3.1. Syntax

To begin with, we will introduce two important elements of our logic: fuzzy modifiers and fuzzy concrete domains.

Fuzzy modifiers. A *fuzzy modifier mod* is a function $f_{mod}\colon [0,1] \to [0,1]$ which applies to a fuzzy set to change its membership function. We will allow modifiers defined in terms of *linear* hedges (Figure 1 (e)) and *triangular* functions (Figure 1 (b)) [25]. Formally:

$$\begin{array}{lrcl} mod \to & \texttt{linear}(c) & | & \text{(M1)} \\ & \texttt{triangular}(a,b,c) & & \text{(M2)} \end{array}$$

where in linear modifiers we assume that $a = c/(c+1), b = 1/(c+1)$.

Example 1. *Modifier* very *can be defined as* $\texttt{linear}(0.8)$.

Fuzzy concrete domains. A *fuzzy concrete domain* [25] (also called a fuzzy *datatype*) $\mathbf{D}$ is a pair $\langle \Delta_{\mathbf{D}}, \Phi_{\mathbf{D}} \rangle$, where $\Delta_{\mathbf{D}}$ is a concrete interpretation domain, and $\Phi_{\mathbf{D}}$ is a set of fuzzy concrete predicates $\mathbf{d}$ with an arity n and an interpretation $\mathbf{d}^{\mathcal{I}} : \Delta_{\mathbf{D}}^n \to [0,1]$, which is an n-ary fuzzy relation over $\Delta_{\mathbf{D}}$.

As fuzzy concrete predicates we allow the following functions defined over an interval $[k_1, k_2] \subseteq \mathbb{Q}$: *trapezoidal* membership function (Figure 1 (a)), the *triangular* (Figure 1 (b)), the *left-shoulder* function (Figure 1 (c)) and the *right-shoulder* function (Figure 1 (d)) [25].

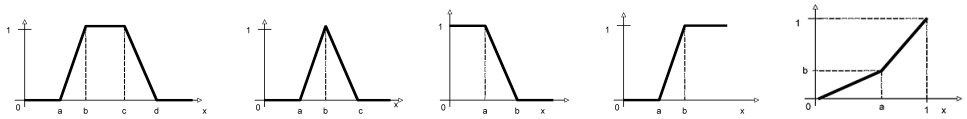
Furthermore, we will also allow *fuzzy modified datatypes*, obtained after the application of a fuzzy modifier *mod* to a fuzzy concrete domain interpretation.

Formally:

$$\begin{array}{lrcl} \mathbf{d} \to & \texttt{left}(k_1,k_2,a,b) & | & \text{(D1)} \\ & \texttt{right}(k_1,k_2,a,b) & | & \text{(D2)} \\ & \texttt{triangular}(k_1,k_2,a,b,c) & | & \text{(D3)} \\ & \texttt{trapezoidal}(k_1,k_2,a,b,c,d) & | & \text{(D4)} \\ & mod(\mathbf{d}) & & \text{(D5)} \end{array}$$

Note that in fuzzy modified datatypes $k_1 = 0, k_2 = 1$. Furthermore, we allow nesting of modifiers, as for example $mod(mod(d))$.

Example 2. *We may define the fuzzy datatype* YoungAge$\colon [0,200] \to [0,1]$*, denoting the degree of a person being young, as* $\mathsf{YoungAge}(x) = \texttt{left}(0, 200, 10, 30)$.



(a) (b) (c) (d) (e)

Figure 1: (a) Trapezoidal function; (b) Triangular function; (c) L-function; (d) R-function; (e) Linear function.

Symbols. Fuzzy $\mathcal{SROIQ}(\mathbf{D})$ assumes three alphabets of symbols, for (abstract and concrete) *fuzzy concepts*, *fuzzy roles* and *individuals*. The syntax of fuzzy concepts and roles is shown in Table 2.

Concept constructors (C1)–(C16) correspond to the concept constructors of crisp $\mathcal{SROIQ}(\mathbf{D})$. The only difference here are modified concepts (C17).

Example 3. *Concept* $\mathsf{Human} \sqcap \exists \mathsf{hasAge}.\mathsf{YoungAge}$ *denotes the fuzzy set of young humans.* $\mathsf{very}(\mathsf{Human} \sqcap \exists \mathsf{hasAge}.\mathsf{YoungAge})$ *denotes* very *young humans.*

Role constructors (R1)–(R3) correspond to the role constructors of crisp $\mathcal{SROIQ}(\mathbf{D})$. (R4) corresponds to modified roles.

Notation. Let us introduce some notation that will be used in the rest of the paper:

- C, D are (possibly complex) fuzzy concepts,
- A is an atomic fuzzy concept,
- R is a (possibly complex) abstract fuzzy role,
- R_A is an atomic fuzzy role,
- S is a simple fuzzy role [1],
- T is a concrete fuzzy role,
- a, b are abstract individuals, v is a concrete individual,
- $\mathbf{d}$ is a fuzzy concrete predicate,
- n, m are natural numbers with $n \geq 0, m > 0$,
- *mod* is a fuzzy modifier,
- $\rhd \in \{\geq, >\}, \lhd \in \{\leq, <\}, \bowtie \in \{\geq, >, \leq, <\}$,
- $\alpha \in [0, 1]$.

A *Fuzzy Knowledge Base* (KB) contains a finite number of axioms. The axioms that are allowed in our logic are shown in Table 2. They can be grouped into a fuzzy ABox with axioms (A1)–(A7), a fuzzy TBox with axioms (A8)–(A11), and a fuzzy RBox with axioms (A12)–(A25). All the axioms have a equivalent in crisp $\mathcal{SROIQ}(\mathbf{D})$.

[1] Simple roles are needed to guarantee the decidability of the logic. Intuitively, simple roles cannot take part in cyclic role inclusion axioms (see [3] for a formal definition).

Example 4. *The fuzzy concept assertion* $\langle \mathsf{paul}\colon \mathsf{Tall} \geq 0.5\rangle$ *states that Paul is tall with at least degree* 0.5*. The fuzzy RIA* $\langle \mathsf{isFriendOf}\ \mathsf{isFriendOf} \sqsubseteq \mathsf{isFriendOf} \geq 0.75\rangle$ *states that the friends of my friends can also be considered as my friends with at least degree* 0.75*.*

3.2. Semantics

Fuzzy interpretation. A fuzzy interpretation $\mathcal{I}$ with respect to a fuzzy concrete domain $\mathbf{D}$ is a pair $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consisting of a non empty set $\Delta^{\mathcal{I}}$ (the interpretation domain) disjoint with $\Delta_{\mathbf{D}}$ and a fuzzy interpretation function $\cdot^{\mathcal{I}}$ mapping:

- A fuzzy *abstract individual* a onto an element $a^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$.
- A fuzzy *concrete individual* v onto an element $v_{\mathbf{D}} \subseteq \Delta_{\mathbf{D}}$.
- A fuzzy *concept* C onto a function $C^{\mathcal{I}} : \Delta^{\mathcal{I}} \to [0, 1]$.
- A fuzzy *abstract role* R onto a function $R^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \to [0, 1]$.
- A fuzzy *concrete role* T onto a function $T^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta_{\mathbf{D}} \to [0, 1]$.
- An n-ary fuzzy *concrete domain* $\mathbf{d}$ onto a function $\mathbf{d}^{\mathcal{I}} : \Delta^n_{\mathbf{D}} \to [0, 1]$.
- A fuzzy *modifier mod* onto a function $f_{mod} : [0, 1] \to [0, 1]]$.

$C^{\mathcal{I}}$ (resp. $R^{\mathcal{I}}$) denotes the membership function of the fuzzy concept C (resp. fuzzy role R) w.r.t. $\mathcal{I}$. $C^{\mathcal{I}}(a)$ (resp. $R^{\mathcal{I}}(a, b)$) gives us to what extent the individual a can be considered as an element of the fuzzy concept C (resp. to what extent (a, b) can be considered as an element of the fuzzy role R) under the fuzzy interpretation $\mathcal{I}$.

The fuzzy interpretation function is defined for fuzzy concepts, roles, concrete domains and axioms as shown in Table 2. We say that a fuzzy interpretation $\mathcal{I}$ satisfies a fuzzy KB $\mathcal{K}$ iff $\mathcal{I}$ satisfies each element in $\mathcal{K}$.

Note that we have included some syntactic sugar axioms: concept equivalences (A9), disjoint concept axioms (A10), disjoint union concepts (A11), domain role axioms (A16), range role restrictions (A17), and functional role axioms (A18). In fact, while in the classical case the meaning of these axioms is very clear, in the fuzzy case this is not always the case. As discussed in [22], there could be alternative definitions for disjoint concepts, and range role axioms. Consequently, it was convenient to write the formal definition of these axioms.

3.3. Reasoning tasks

There are several reasoning tasks in fuzzy $\mathcal{SROIQ}(\mathbf{D})$ [23, 26].

- *Fuzzy KB satisfiability.* A fuzzy interpretation $\mathcal{I}$ *satisfies* (is a model of) a fuzzy KB $\mathcal{K}$ iff it satisfies each axiom in $\mathcal{K}$.
- *Concept satisfiability.* C is α-satisfiable w.r.t. a fuzzy KB $\mathcal{K}$ iff there exists a model $\mathcal{I}$ of $\mathcal{K}$ such that $C^I(x) \geq \alpha$ for some $x \in \Delta^{\mathcal{I}}$.

Table 2: Syntax and semantics of the fuzzy DL $\mathcal{SROIQ}(\mathbf{D})$.

Concept	**Syntax** (C)	**Semantics of** $C^{\mathcal{I}}(x)$
(C1)	A	$A^{\mathcal{I}}(x)$
(C2)	$\top$	1
(C3)	$\bot$	0
(C4)	$C \sqcap D$	$C^{\mathcal{I}}(x) \otimes D^{\mathcal{I}}(x)$
(C5)	$C \sqcup D$	$C^{\mathcal{I}}(x) \oplus D^{\mathcal{I}}(x)$
(C6)	$\neg C$	$\ominus C^{\mathcal{I}}(x)$
(C7)	$\forall R.C$	$\inf_{y \in \Delta^{\mathcal{I}}} \{R^{\mathcal{I}}(x,y) \Rightarrow C^{\mathcal{I}}(y)\}$
(C8)	$\exists R.C$	$\sup_{y \in \Delta^{\mathcal{I}}} \{R^{\mathcal{I}}(x,y) \otimes C^{\mathcal{I}}(y)\}$
(C9)	$\forall T.\mathbf{d}$	$\inf_{v \in \Delta_{\mathbf{D}}} \{T^{\mathcal{I}}(x,v) \Rightarrow \mathbf{d}^{\mathcal{I}}(v)\}$
(C10)	$\exists T.\mathbf{d}$	$\sup_{v \in \Delta_{\mathbf{D}}} \{T^{\mathcal{I}}(x,v) \otimes \mathbf{d}^{\mathcal{I}}(v)\}$
(C11)	$\{\alpha/a\}$	α if $x = o_i^{\mathcal{I}}$, 0 otherwise
(C12)	$\geq m\ S.C$	$\sup_{y_1,\ldots,y_m \in \Delta^{\mathcal{I}}} (\min_{i=1}^{m} \{S^{\mathcal{I}}(x,y_i) \otimes C^{\mathcal{I}}(y_i)\}) \bigotimes ((\otimes)_{1 \leq j < k \leq m} \{y_j \neq y_k\})$
(C13)	$\leq n\ S.C$	$\inf_{y_1,\ldots,y_{n+1} \in \Delta^{\mathcal{I}}} (\min_{i=1}^{n+1} \{S^{\mathcal{I}}(x,y_i) \otimes C^{\mathcal{I}}(y_i)\}) \Rightarrow ((\oplus)_{1 \leq j < k \leq n+1} \{y_j = y_k\})$
(C14)	$\geq m\ T.\mathbf{d}$	$\sup_{v_1,\ldots,v_m \in \Delta_{\mathbf{D}}} (\min_{i=1}^{m} \{T^{\mathcal{I}}(x,v_i) \otimes \mathbf{d}^{\mathcal{I}}(v_i)\}) \bigotimes ((\otimes)_{j<k} \{v_j \neq v_k\})$
(C15)	$\leq n\ T.\mathbf{d}$)	$\inf_{v_1,\ldots,v_{n+1} \in \Delta_{\mathbf{D}}} (\min_{i=1}^{n+1} \{T^{\mathcal{I}}(x,v_i) \otimes \mathbf{d}^{\mathcal{I}}(v_i)\}) \Rightarrow ((\oplus)_{j<k} \{v_j = v_k\})$
(C16)	$\exists S.\texttt{Self}$	$S^{\mathcal{I}}(x,x)$
(C17)	$mod(C)$	$f_{mod}(C^{\mathcal{I}}(x))$
(C18)	$\alpha \cdot C$	$\alpha \cdot C^{\mathcal{I}}(x)$
(C19)	$(\alpha_1 \cdot C_1) + \cdots + (\alpha_k \cdot C_k)$	$\sum_{i=1}^{k} \alpha_i \cdot C_i^{\mathcal{I}}(x)$
Role	**Syntax** (R)	**Semantics of** $R^{\mathcal{I}}(x,y)$
(R1)	R_A	$R_A^{\mathcal{I}}(x,y)$
(R2)	R^-	$R^{\mathcal{I}}(y,x)$
(R3)	U	1
(R4)	$mod(R)$	$f_{mod}(R^{\mathcal{I}}(x,y))$
(R5)	T	$T^{\mathcal{I}}(x,y)$
Datatype	**Syntax (d)**	**Semantics of** $\mathbf{d}^{\mathcal{I}}$
(D1–D4)	See Section 3.1	$\mathbf{d_D}$
(D5)	$mod(\mathbf{d})$	$f_{mod}(\mathbf{d}^{\mathcal{I}})$
Axiom	**Syntax** (τ)	**Semantics** ($\mathcal{I}$ satisfies τ if ...)
(A1)	$\langle a\!:\!C \bowtie \alpha \rangle$	$C^{\mathcal{I}}(a^{\mathcal{I}}) \bowtie \alpha$
(A2)	$\langle (a,b)\!:\!R \bowtie \alpha \rangle$	$R^{\mathcal{I}}(a^{\mathcal{I}}, b^{\mathcal{I}}) \bowtie \alpha$
(A3)	$\langle (a,b)\!:\!\neg R \bowtie \alpha \rangle$	$\ominus R^{\mathcal{I}}(a^{\mathcal{I}}, b^{\mathcal{I}}) \bowtie \alpha$
(A4)	$\langle (a,v)\!:\!T \bowtie \alpha \rangle$	$T^{\mathcal{I}}(a^{\mathcal{I}}, v_{\mathbf{D}}) \bowtie \alpha$
(A5)	$\langle (a,v)\!:\!\neg T \bowtie \alpha \rangle$	$\ominus T^{\mathcal{I}}(a^{\mathcal{I}}, v_{\mathbf{D}}) \bowtie \alpha$
(A6)	$\langle a \neq b \rangle$	$a^{\mathcal{I}} \neq b^{\mathcal{I}}$
(A7)	$\langle a = b \rangle$	$a^{\mathcal{I}} = b^{\mathcal{I}}$
(A8)	$\langle C \sqsubseteq D \rhd \alpha \rangle$	$\inf_{x \in \Delta^{\mathcal{I}}} \{C^{\mathcal{I}}(x) \Rightarrow D^{\mathcal{I}}(x)\} \rhd \alpha$
(A9)	$C_1 \equiv \ldots C_m$	$\forall_{x \in \Delta^{\mathcal{I}}} C_1^{\mathcal{I}}(x) = \cdots = C_m^{\mathcal{I}}(x)$
(A10)	$\texttt{dis}(C_1, \ldots, C_m)$	$\forall x, y \in \Delta^{\mathcal{I}}, \min\{C_1^{\mathcal{I}}(x,y), \ldots, C_m^{\mathcal{I}}(x,y)\} = 0$
(A11)	$\texttt{disUnion}(C_1, \ldots, C_m)$	$\texttt{dis}(C_2, \ldots, C_m), C_1 \equiv C_2 \sqcup \cdots \sqcup C_m$
(A12)	$\langle R_1 \ldots R_m \sqsubseteq R \rhd \alpha \rangle$	$\inf_{x_1, x_{n+1} \in \Delta^{\mathcal{I}}} \{\sup_{x_2 \ldots x_m \in \Delta^{\mathcal{I}}} \{(R_1^{\mathcal{I}}(x_1,x_2) \otimes \cdots \otimes R_n^{\mathcal{I}}(x_m, x_{m+1})) \Rightarrow R^{\mathcal{I}}(x_1, x_{m+1})\}\} \rhd \alpha$
(A13)	$\langle T_1 \sqsubseteq T_2 \rhd \alpha \rangle$	$\inf_{x \in \Delta^{\mathcal{I}}, v \in \Delta_{\mathbf{D}}} \{T_1^{\mathcal{I}}(x,v) \Rightarrow T_2^{\mathcal{I}}(x,v)\} \rhd \alpha$
(A14)	$R_1 \equiv \ldots R_m$	$\forall_{x,y \in \Delta^{\mathcal{I}}} R_1^{\mathcal{I}}(x,y) = \cdots = R_m^{\mathcal{I}}(x,y)$
(A15)	$T_1 \equiv \ldots T_m$	$\forall_{x \in \Delta^{\mathcal{I}}, v \in \Delta_{\mathbf{D}}} R_1^{\mathcal{I}}(x,v) = \cdots = R_m^{\mathcal{I}}(x,v)$
(A16)	$\texttt{domain}(R, C)$	$\langle \exists R.\top \sqsubseteq C \geq 1 \rangle$
(A17)	$\texttt{range}(R, C)$	$\langle \top \sqsubseteq \forall R.C \geq 1 \rangle$
(A18)	$\texttt{func}(R)$	$\langle \top \sqsubseteq (\leq 1\ R.\top) \geq 1 \rangle$
(A19)	$\texttt{trans}(R)$	$\forall x, y, z \in \Delta^{\mathcal{I}}, R^{\mathcal{I}}(x,z) \otimes R^{\mathcal{I}}(z,y) \leq R^{\mathcal{I}}(x,y)$
(A20)	$\texttt{dis}(S_1, \ldots, S_m)$	$\forall x, y \in \Delta^{\mathcal{I}}, \min\{S_1^{\mathcal{I}}(x,y), \ldots, S_m^{\mathcal{I}}(x,y)\} = 0$
(A21)	$\texttt{dis}(T_1, \ldots, T_m)$	$\forall x \in \Delta^{\mathcal{I}}, v \in \Delta_{\mathbf{D}}, \min\{T_1^{\mathcal{I}}(x,v), \ldots, T_m^{\mathcal{I}}(x,v)\} = 0$
(A22)	$\texttt{ref}(R)$	$\forall x \in \Delta^{\mathcal{I}}, R^{\mathcal{I}}(x,x) = 1$
(A23)	$\texttt{irr}(S)$	$\forall x \in \Delta^{\mathcal{I}}, S^{\mathcal{I}}(x,x) = 0$
(A24)	$\texttt{sym}(R)$	$\forall x, y \in \Delta^{\mathcal{I}}, R^{\mathcal{I}}(x,y) = R^{\mathcal{I}}(y,x)$
(A25)	$\texttt{asy}(S)$	$\forall x, y \in \Delta^{\mathcal{I}}$, if $S^{\mathcal{I}}(x,y) > 0$ then $S^{\mathcal{I}}(y,x) = 0$

- *Entailment*: A fuzzy concept (or role) assertion τ is entailed by a fuzzy KB $\mathcal{K}$ iff every model of $\mathcal{K}$ satisfies τ.
- *Concept subsumption*: D subsumes C (denoted $C \sqsubseteq D$) w.r.t. a fuzzy KB $\mathcal{K}$ iff every model $\mathcal{I}$ of $\mathcal{K}$ satisfies $\forall x \in \Delta^{\mathcal{I}}, C^{\mathcal{I}}(x) \leq D^{\mathcal{I}}(x)$.
- *Best degree bound* (BDB). The BDB of a concept or role assertion τ is defined as the $\sup\{\alpha : \mathcal{K} \models \langle \tau \geq \alpha \rangle\}$.
- *Maximal concept satisfiability degree*. The maximal satisfiability degree of a fuzzy concept C w.r.t. a fuzzy KB $\mathcal{K}$ is defined as the $\sup\{\alpha | C$ is α-satisfiable $\}$.

However, these reasoning tasks are part of the query language and not of the representation language. Thus, we shall not represent them in a fuzzy ontology.

4. Representation of Fuzzy Ontologies in OWL 2

In this section we will explain a methodology to represent fuzzy $\mathcal{SROIQ}(\mathbf{D})$ ontologies using OWL 2. We anticipate that the methodology has some differences with a previous version in the paper [8], as explained in Section 6.

The idea of our representation is to use an OWL 2 ontology, extending their elements with annotation properties representing the features of the fuzzy ontology that OWL 2 cannot directly encode.

For the sake of clarity, we will use OWL 2 abstract syntax [17] for OWL 2, and an XML syntax to write the value of annotation properties[2].

Let us begin with an illustrating example.

Example 5. *Consider the fuzzy concept assertion of Example 4,* $\langle$*paul*: *Tall* $\geq 0.5\rangle$. *To represent it in OWL 2, we consider the crisp assertion* *paul*: *Tall as represented in OWL 2,* `ClassAssertion(paul Tall)` *and then we add an annotation property including the information* ≥ 0.5 *to it.*

It is worth to note that OWL 2 only provides for annotations on ontologies, axioms, and entities [17]. This is not the case of OWL DL, which only provides for annotations on ontologies and entities.

4.1. Syntactic Requirements of Fuzzy Ontologies

To begin with, we will summarize the syntactic differences between the fuzzy and non-fuzzy ontologies. There are 6 cases depending on the annotated element.

Case 1. Fuzzy modifiers do not have an equivalence in the non-fuzzy case: (M1), (M2).

[2]Of course, the final result depends on the syntax (for instance, in OWL 2 XML syntax the characters $\geq$ and $\leq$ of the annotations are escaped), but OWL 2 ontology editors make these issues transparent to the user.

Case 2. Fuzzy datatypes do not have an equivalence in the non-fuzzy case: (D1)–(D5).

Case 3. Some fuzzy concepts have syntactic differences with the non-fuzzy case (C11) or do not have an equivalence (C17)–(C19).

Case 4. Some fuzzy roles do not have an equivalence in the non-fuzzy case: (R4).

Case 5. Some axioms require an inequality sign and a degree of truth: (A1)–(A5), (A8), (A12)–(A13).

Case 6. Ontologies can be annotated with a fuzzy logic.

4.2. Annotations

Instead of using any of the defaults annotation properties from OWL 2, we will use an annotation property `fuzzyLabel`. Furthermore, for every element of the ontology there can be at-most one annotation of this type.

Every annotation will be delimited by a start tag `<FuzzyOwl2>` and an end tag `</FuzzyOwl2>`, with an attribute `fuzzyType` specifying the fuzzy element being tagged. In the following, we will address the different cases in detail.

4.3. Fuzzy modifiers

According to Section 3.1, the fuzzy modifiers that we want to represent have parameters a, b, c. Consequently, they can be represented as in the previous case, with the particularities that the type of datatype should be double (`xsd:double`) and that there is no need to use `xsd:minInclusive` and `xsd:maxInclusive` (they are assumed to be $0, 1$).

The value of `fuzzyType` will be `modifier`, and there will be a tag `Modifier` with an attribute `type` (possible values `linear`, and `triangular`), and attributes `a`, `b`, `c`, depending on the type of the modifier.

Domain of the annotation. An OWL 2 datatype declaration of the type base double `xsd:double`.

Syntax for the annotation.

```
<fuzzyOwl2 fuzzyType="modifier">
  <MODIFIER>
</fuzzyOwl2>

<MODIFIER> :=
  <Modifier type="linear" c="<DOUBLE>" /> |
  <Modifier type="triangular" a="<DOUBLE>" b="<DOUBLE>" c="<DOUBLE>" />
```

Semantical restrictions. The parsers should check that the following constraints:

- $a, b, c \in [0, 1]$
- $b = 0$ iff $a = 1$
- $b = 1$ iff $c = 1$

Example 6. *Let us define the fuzzy modifier* Very $=$ `linear`(0.8). *We create a datatype* Very.

```
DatatypeDefinition ( Very DatatypeRestriction (
  xsd:double
  xsd:minInclusive "0"^^xsd:double
  xsd:maxInclusive "1"^^xsd:double
) )
```

Then, we add the following annotation property to it:

```
<fuzzyOwl2 fuzzyType="modifier">
  <Modifier type="linear" c="0.8" />
</fuzzyOwl2>
```

4.4. Fuzzy datatypes

Firstly, we will consider fuzzy datatypes (D1)–(D4), and then we will consider the case (D5).

4.4.1. Fuzzy atomic datatypes

According to Section 3.1, these fuzzy datatypes have parameters k_1, k_2, a, b, c, d. The first four parameters are common to all of them, c only appears in (D4), (D5); and d only appears in (D5).

Domain of the annotation. An OWL 2 datatype declaration of the type base of the fuzzy datatype (integer `xsd:integer` or double `xsd:double`), such that:

```
xsd:minInclusive="<DOUBLE>"
xsd:maxInclusive="<DOUBLE>"
```

`<DOUBLE>` denotes a rational number. `xsd:minInclusive` should take the value k_1, whereas `xsd:maxInclusive` should take the value k_2. These parameters are optional and, if omitted, then the minimum and maximum of the attributes (a, b, c, d) is assumed, respectively.

Syntax for the annotation.

```
<fuzzyOwl2 fuzzyType="datatype">
  <DATATYPE>
</fuzzyOwl2>

<DATATYPE> :=
  <Datatype type="leftshoulder" a="<DOUBLE>" b="<DOUBLE>" /> |
  <Datatype type="rightshoulder" a="<DOUBLE>" b="<DOUBLE>" /> |
  <Datatype type="triangular" a="<DOUBLE>" b="<DOUBLE>" c="<DOUBLE>" /> |
  <Datatype type="trapezoidal" a="<DOUBLE>" b="<DOUBLE>" c="<DOUBLE>" d="<DOUBLE>" />
```

Semantical restrictions. The parsers should check the following restrictions:

- $k_1 \leq a \leq b \leq c \leq d \leq k_2$ is verified.

Example 7. *Let us represent the fuzzy datatype* YoungAge $=$ `left`$(0, 200, 10, 30)$ *denoting the age of a young person. This fuzzy datatype is represented using a datatype definition of base type* `xsd:integer` *with range in* $[0, 200]$*:*

```
DatatypeDefinition ( YoungAge DatatypeRestriction (
  xsd:integer
  xsd:minInclusive "0"^^xsd:integer
  xsd:maxInclusive "200"^^xsd:integer
) )
```

Then we add the following annotation property to it:

```
<fuzzyOwl2 fuzzyType="datatype">
  <Datatype type="leftshoulder" a="10" b="30" />
</fuzzyOwl2>
```

4.4.2. Fuzzy modified datatypes

In this case, the parameters are two: the modifier, and the fuzzy datatype that is being modified.

Domain of the annotation. An OWL 2 datatype declaration of any type base.

Syntax for the annotation.

```
<fuzzyOwl2 fuzzyType="datatype">
  <Datatype type="modified" modifier="<STRING>" base="<STRING>" />
</fuzzyOwl2>
```

Semantical restrictions. The parsers should check the following restrictions:

- *modifier* has already been defined as a fuzzy modifier.
- *base* has already been defined as a fuzzy datatype.

Example 8. *Let us represent the fuzzy datatype* VeryYoungAge*. To begin with, we assume that the fuzzy datatype* YoungAge *has been created as in Example 7, and that the fuzzy datatype* very *has been created as in Example 6. Next, we define a new datatype* VeryYoungAge*, adding the following annotation property to it:*

```
<fuzzyOwl2 fuzzyType="datatype">
  <Datatype type="modified" modifier="very" base="YoungAge" />
</fuzzyOwl2>
```

4.5. Fuzzy concepts

In this case, we create a new concept D and to add an annotation property describing the type of the constructor and the value of their parameters. Now, the value of `fuzzyType` is `concept`, and there is a tag `Concept` with an attribute `type`, and other attributes, depending on the concept constructor. The general rule is that recursion is not allowed, i.e., D cannot be defined in terms of D, so D is not a valid value for these attributes.

4.5.1. Fuzzy modified concepts

Here, the value of `type` is `modified`. There are also two additional attributes: `modifier` (fuzzy modifier), and `base` (the name of the fuzzy concept that is being modified).

Domain of the annotation. An OWL 2 concept declaration.

Syntax for the annotation.

```
<fuzzyOwl2 fuzzyType="concept">
  <MODIFIED_CONCEPT>
</fuzzyOwl2>

<MODIFIED_CONCEPT> := <Concept type="modified" modifier="<STRING>"
base="<STRING>" />
```

Semantical restrictions. The parsers should check the following restrictions:

- *modifier* has already been defined as a fuzzy modifier.
- The name of the concept C is different from the name of the annotated concept.

Example 9. *Let us represent now the concept* $\mathsf{very}(C)$*. We assume that the fuzzy modifier has been created as in Example 6. To that end, we create the atomic concept* VeryC *and annotate it:*

```
Class ( VeryC Annotation( fuzzyLabel
  <fuzzyOwl2 fuzzyType="concept">
    <Concept type="modified" modifier="very" base="C" />
  </fuzzyOwl2>
) )
```

4.5.2. Weighted concepts

Here, the value of `type` is `weighted`. There are also two additional attributes: `value` (a real number in $(0, 1]$), and `base` (the name of the fuzzy concept that is being weighted).

Domain of the annotation. An OWL 2 concept declaration.

Syntax for the annotation.

```
<fuzzyOwl2 fuzzyType="concept">
  <WEIGHTED_CONCEPT>
</fuzzyOwl2>

<WEIGHTED_CONCEPT> := <Concept type="weighted" value="<DOUBLE>" base="<STRING>" />
```

Semantical restrictions. The parsers should check the following restrictions:

- *value* in $(0, 1]$.
- The name of the concept C is different from the name of the annotated concept.

Example 10. *Let us represent now the concept* $(0.8\ C)$. *We create the atomic* Weight0.8C *and annotate it:*

```
Class ( Weight0.8C Annotation( fuzzyLabel
  <fuzzyOwl2 fuzzyType="concept">
    <Concept type="weighted" value="0.8" base="C" />
  </fuzzyOwl2>
) )
```

4.5.3. Weighted sum concepts

Here, the value of `type` is `weightedSum`. There are also several additional tags representing weighted concepts.

Domain of the annotation. An OWL 2 concept declaration.

Syntax for the annotation.

```
<fuzzyOwl2 fuzzyType="concept">
  <Concept type="weightedSum">
    (<WEIGHTED_CONCEPT>)+
  </Concept>
</fuzzyOwl2>
```

Semantical restrictions. Let k be the number of weighted concepts taking part in the definition. The parsers should check the following restrictions:

- $k \geq 2$.
- $\sum_{i=1}^{k} value_k = 1$.
- The names of the concepts C_i are different from the name of the annotated concept.

Example 11. *Let us represent now the concept* $(0.8\ A + 0.2\ B)$. *We create the atomic* Sum08Aplus02B *and annotate it:*

```
Class ( Sum08Aplus02B Annotation( fuzzyLabel
  <fuzzyOwl2 fuzzyType="concept">
    <Concept type="weightedSum">
      <Concept type="weighted" value="0.8" base="A" />
      <Concept type="weighted" value="0.2" base="B" />
    </Concept>
  </fuzzyOwl2>
) )
```

4.5.4. Fuzzy nominals

Here, the value of `type` is `nominal`. There are also two additional attributes: `value` (a real number in $(0, 1]$), and `individual` (the name of the individual that is being weighted).

Domain of the annotation. An OWL 2 concept declaration.

Syntax for the annotation.

```
<fuzzyOwl2 fuzzyType="concept">
  <FUZZY_NOMINAL_CONCEPT>
</fuzzyOwl2>

<FUZZY_NOMINAL_CONCEPT> := <Concept type="nominal" value=<DOUBLE> individual=<STRING> />
```

Semantical restrictions. The parsers should check the following restrictions:

- *value* $\in (0, 1]$.

Example 12. *Let us represent now the concept* $\{0.75/\mathsf{ind}\}$. *We create the atomic* ind075 *and annotate it:*

```
Class ( ind075 Annotation( fuzzyLabel
  <fuzzyOwl2 fuzzyType="concept">
    <Concept type="nominal" value="0.75" individual="ind" />
  </fuzzyOwl2>
) )
```

4.6. Fuzzy roles

In this case, we create a new concept R and to add an annotation property describing the type of the constructor and the value of their parameters. Now, the value of `fuzzyType` is `role`, and there is a tag `Role` with an attribute `type`, and other attributes, depending on the role constructor. The general rule is that recursion is not allowed. For the moment, we only support fuzzy modified roles.

4.6.1. Fuzzy modified roles

Here, the value of `type` is `modified`. There are also two additional attributes: `modifier` (fuzzy modifier), and `base` (the name of the fuzzy role that is being modified).

Domain of the annotation. An OWL 2 (object or data) property declaration.

Syntax for the annotation.

```
<fuzzyOwl2 fuzzyType="role">
  <MODIFIED_ROLE>
</fuzzyOwl2>

<MODIFIED_ROLE> := <Role type="modified" modifier="<STRING>"
base="<STRING>" />
```

Semantical restrictions. The parsers should check the following restrictions:

- *modifier* has already been defined as a fuzzy modifier.
- The name of the role R is different from the name of the annotated concept.

Example 13. *Let us represent now the abstract role* $\mathsf{very}(R)$. *We assume that the fuzzy modifier has been created as in Example 6. To that end, we create the atomic object property* VeryR *and annotate it:*

```
ObjectProperty ( VeryR Annotation( fuzzyLabel
  <fuzzyOwl2 fuzzyType="role">
    <Role type="modified" modifier="very" base="R" />
  </fuzzyOwl2>
) )
```

4.7. Fuzzy axioms

It is possible to add a degree of truth to some axioms, i.e., (A1)–(A5), (A8), (A12)–(A13). The value of `fuzzyType` is `axiom`. There is an optional tag `Degree`, with and attribute `value`. If omitted, we assume degree 1.

It would also be possible to specify an inequality sign but we will assume $\geq$. An axiom of the form $\langle \tau > \alpha \rangle$ is equivalent to $\langle \tau \geq \alpha + \epsilon \rangle$. Regarding axioms involving $\lhd$, note that $\langle \tau \lhd \alpha \rangle$ is equivalent to $\langle \tau \lhd^{-} 1 - \alpha \rangle$[3] in axioms (A1)–(A5). In axioms $(A8), (A12), (A13)$ we argue that it does not make sense to have axioms of the form $\langle \tau \lhd \alpha \rangle$ because such axioms do not have an equivalent expression in classical DLs.

Domain of the annotation. An OWL 2 axiom of the following types: concept assertion,role assertion, GCI, RIA. That is, the crisp equivalents of axioms (A1), (A1)–(A5), (A8), (A12)–(A13).

Syntax for the annotation.

```
<fuzzyOwl2 fuzzyType="axiom">
  <Degree value="<DOUBLE>" />
</fuzzyOwl2>
```

Semantical restrictions. The parsers should check the following restrictions:

- *value* in $(0, 1]$.

Example 14. *Let us consider again, in greater detail, Example 5. Firstly, we create an OWL 2 concept assertion:*

```
ClassAssertion(paul Tall)
```

Then, we annotate it as follows:

```
<fuzzyOwl2 fuzzyType="axiom">
  <Degree value="0.5" />
</fuzzyOwl2>
```

[3] $\bowtie^{-}$ denotes the reflection of the operator $\bowtie$ and is defined as follows: $\geq^{-} = \leq, >^{-} = <, \leq^{-} = \geq <^{-} = >$.

4.8. Ontologies

We may also annotate the ontology and specify the fuzzy logic to be considered in the semantics.

The value of `fuzzyType` is `ontology`. There is a tag `FuzzyLogic`, with and attribute `logic`, that specifies the default fuzzy logic which is used in the semantics of the fuzzy ontology.

Domain of the annotation. An OWL 2 ontology.

Syntax for the annotation.

```
<fuzzyOwl2 fuzzyType="ontology">
  <FuzzyLogic logic=<LOGIC> />
</fuzzyOwl2>

<LOGIC> := "lukasiewicz" | "zadeh"
```

At the moment, we only allow two fuzzy logics, Łukasiewicz and Zadeh.

5. Some Applications of Fuzzy Ontologies

In this section, we will provide some examples illustrating how use fuzzy ontologies to model the knowledge in real application problems, and how to encode the fuzzy ontologies using the methodology explained in Section 4[4].

5.1. Matchmaking

To begin with, we will address the family of matchmaking problems. The following example is a modified version of the one in [5].

Assume that a car seller sells a sedan car. A buyer is looking for a second hand passenger car. Both the buyer as well as the seller have preferences (restrictions). Our aim is to find the best agreement. The preferences are as follows. Concerning the buyer:

1. If there is an alarm system in the car then he is completely satisfied with paying no more than 22300, but he can go up to 22750 to a lesser degree of satisfaction.
2. He wants a driver insurance and either a theft insurance or a fire insurance.
3. He wants air conditioning and the external color should be either black or grey.
4. Preferably the price is no more than 22000, but he can go up to 24000 to a lesser degree of satisfaction.

[4]The full examples may be downloaded from `http://www.straccia.info`.

5. The kilometer warranty is preferably at least 175000, but he may go down to 150000 to a lesser degree of satisfaction.

6. The weights of the preferences 1–5 are $0.1, 0.2, 0.1, 0.2, 0.4$, respectively. The higher the value, the more important the preference is.

7. There is a strict requirement: he does not want to pay more than 26000 (buyer reservation value).

Concerning the seller:

1. If there is an navigator pack system in the car then he is completely satisfied with a price of at least 22750, but he can go down to 22500 to a lesser degree of satisfaction.

2. He would prefer to sell the Insurance Plus package.

3. The kilometer warranty is preferably at most 100000, but he may go up to 125000 to a lesser degree of satisfaction.

4. The monthly warranty is preferably at most 60, but he may go up to 72 to a lesser degree of satisfaction.

5. If the color is black then the car has air conditioning.

6. The weights of the preferences 1–5 are, $0.3, 0.1, 0.3, 0.1, 0.2$, respectively. The higher the value, the more important the preference is.

7. There is a strict requirement: he wants to sell no less than 22000 (seller reservation value).

We have also some background theory about the domain:

1. There are several types of vehicles: car, sport utility vehicle (SUV), truck, and van. Each of these vehicles has some subclasses. For instance, there are luxury cars and passenger cars. In particular, a sedan is a passenger car (see Figure 2).

2. There are several car makers, e.g., BMW, Ferrari, Volkswagen ...

3. There are several car colors, e.g., back, grey ...

4. A satellite alarm system is an alarm system.

5. The *Navigator Pack* is a satellite alarm system with a GPS system.

6. The *Insurance Plus Package* is a driver insurance together with a theft insurance.

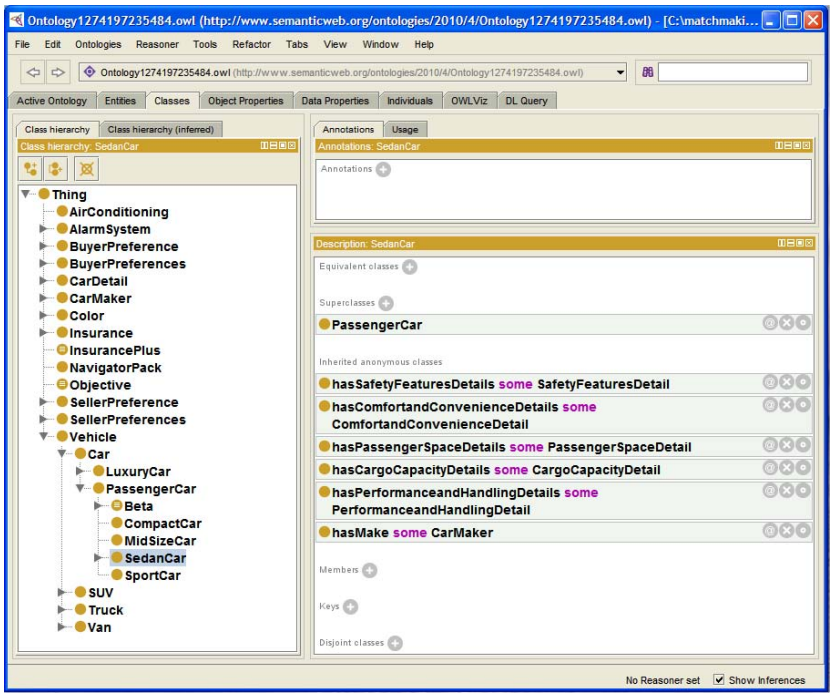


Figure 2: Definition of the concept Sedan.

Let us show now how to encode the previous knowledge. A concept Buy collects all the buyer's preferences together in such a way that the higher is the maximal degree of satisfiability of Buy, the more the buyer is satisfied.

$$\mathsf{Buy} = \mathsf{BuyerRequirements} \sqcap \mathsf{BuyerPreferences}$$
$$\mathsf{BuyerRequirements} = \mathsf{PassengerCar} \sqcap \exists \mathsf{hasPrice.leq26000}$$
$$\mathsf{B1} = \neg(\exists \mathsf{hasAlarmSystem.AlarmSystem}) \sqcup \exists \mathsf{hasPrice.ls22300\text{-}22750}$$
$$\mathsf{B2} = (\exists \mathsf{hasInsurance.DriverInsurance}) \sqcap \exists \mathsf{hasInsurance.}(\mathsf{TheftInsurance} \sqcup \mathsf{FireInsurance})$$
$$\mathsf{B3} = (\exists \mathsf{hasAirConditioning.AirConditioning}) \sqcap \exists \mathsf{HasExColor.}(\mathsf{ExColorBlack} \sqcup \mathsf{ExColorGray})$$
$$\mathsf{B4} = \exists \mathsf{hasPrice.ls22000\text{-}24000}$$
$$\mathsf{B5} = \exists \mathsf{hasKMWarranty.rs15000\text{-}175000}$$

BuyerPreferences is a weighted sum concept, so we add the following annotation property to it:

```
<fuzzyOwl2 fuzzyType="concept">
  <Concept type="weightedSum">
    <Concept type="weighted" value="0.1" base="B1" />
    <Concept type="weighted" value="0.2" base="B2" />
    <Concept type="weighted" value="0.1" base="B3" />
    <Concept type="weighted" value="0.2" base="B4" />
    <Concept type="weighted" value="0.4" base="B5" />
  </Concept>
</fuzzyOwl2>
```

leq26000, ls22300-22750, ls22000-24000, and rs15000-175000 are defined datatypes with annotation properties. For instance, ls22000-24000 has the following annotation property (see Figure 3):

```
<fuzzyOwl2 fuzzyType="datatype">
  <Datatype type="leftshoulder" a="22000" b="24000" />
</fuzzyOwl2>
```

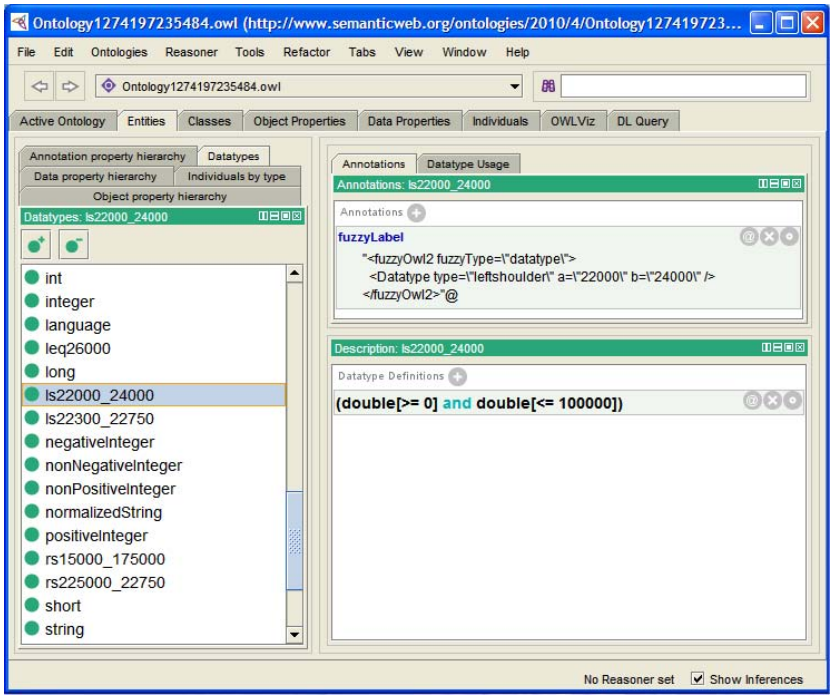


Figure 3: Annotation property defining fuzzy datatype ls22000-24000.

Note that if $a = b$, then we have a crisp concept. This is the case of the datatype leq26000, which is represented as follows:

```
<fuzzyOwl2 fuzzyType="datatype">
  <Datatype type="leftshoulder" a="26000" b="26000" />
</fuzzyOwl2>
```

Similarly to the buyer case, the concept Sell collects all the seller's preferences together in such a way that the higher is the maximal degree of satisfiability of Sell, the more the seller is satisfied.

$\mathsf{Sell} = \mathsf{SellerRequirements} \sqcap \mathsf{SellerPreferences}$
$\mathsf{SellerRequirements} = \mathsf{SedanCar} \sqcap \exists \mathsf{hasPrice.geq22000}$
$\mathsf{S1} = \neg(\exists \mathsf{hasNavigator.NavigatorPack}) \sqcup \exists \mathsf{hasPrice.rs225000\text{-}22750}$
$\mathsf{S2} = \exists \mathsf{hasInsurance.InsurancePlus}$
$\mathsf{S3} = \exists \mathsf{hasKMWarranty.SellerKmWarr}$
$\mathsf{S4} = \exists \mathsf{hasMWarranty.SellerMWarr}$
$\mathsf{S5} = \neg(\exists \mathsf{hasExColor.ExColorBlack}) \sqcup \exists \mathsf{hasAirConditioning.AirConditioning}$

SellerPreferences is a weighted sum concept, so we add the following annotation property to it (see Figure 4):

```
<fuzzyOwl2 fuzzyType="concept">
  <Concept type="weightedSum">
    <Concept type="weighted" value="0.3" base="S1" />
    <Concept type="weighted" value="0.1" base="S2" />
    <Concept type="weighted" value="0.3" base="S3" />
    <Concept type="weighted" value="0.1" base="S4" />
    <Concept type="weighted" value="0.2" base="S5" />
  </Concept>
</fuzzyOwl2>
```

Similar as in the case of the buyer, geq22000, rs225000-22750, SellerKmWarr, SellerMWarr are defined datatypes. For instance, SellerKmWarr is defined as:

```
<fuzzyOwl2 fuzzyType="datatype">
  <Datatype type="leftshoulder" a="100000" b="125000" />
</fuzzyOwl2>
```

Now, it is clear that the best agreement among the buyer and the seller is determined by the maximal degree of satisfiability of the conjunction $\mathsf{Buy} \sqcap \mathsf{Sell}$ under Łukasiewicz fuzzy logic. So, an optimal match (the degree is 0.7625) would be an agreement on a price of 22500, with 100000 kilometer warranty and 60 month warranty.

5.2. Multi-criteria Decision Making

Now, we will concentrate in the family of fuzzy multi-criteria decision making (MCDM) problems. The following example is a modified version of the one in [27].

Given a set of n decision alternatives and a set of m criteria according to which the desirability of an action is judged, a MCDM problem of m criteria and n alternatives consists in determining the optimal alternative $a^\star$ with the highest degree of desirability.

Usually, *alternatives* represent different choices of action available to the decision maker. The decision *criteria* (also referred to as goals or attributes) represent the different dimensions from which the alternatives can be viewed. For instance, cost, quality, or delivery time. A standard feature of MCDM methods is that a MCDM problem can be expressed by means of a *decision matrix*. In the matrix, each row corresponds to an alternative a_i, and each column belongs to a criterion c_j. The score p_{ij} describes the *performance* of alternative a_i against criterion c_j.

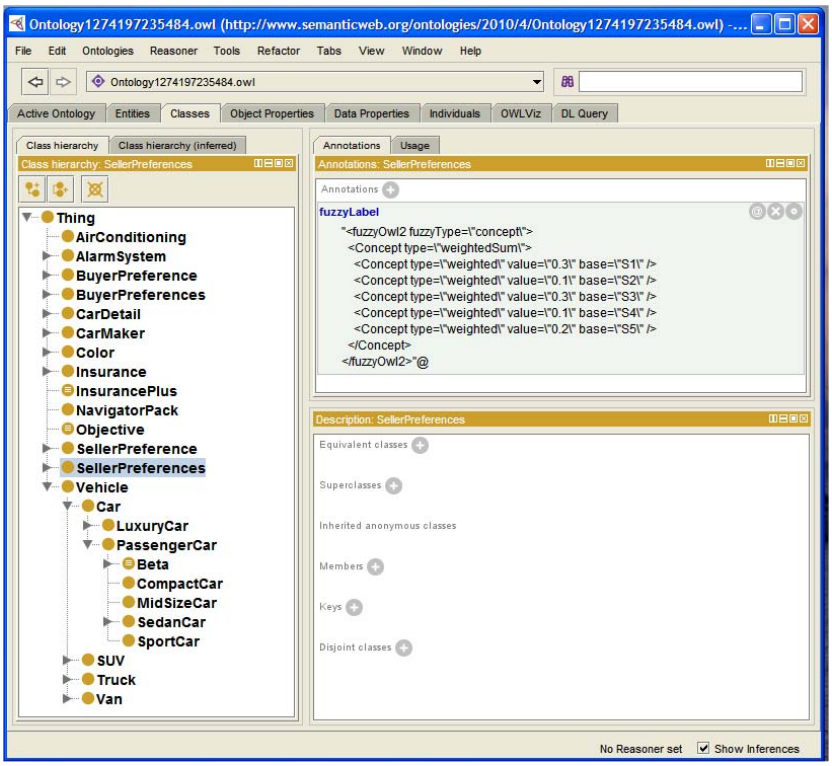


Figure 4: Annotation property defining concept SellerPreferences.

Most of the MCDM methods require to establish the relative importance of every criterion in the decision by assigning a *weight* to it. The weights of the criteria are usually determined on subjective basis and may also be seen as a kind of profit of the criteria. Usually, these weights are normalized to add up to one.

We assume the existence of some *experts* e_k that define the performances and the weights. Given a criterion c_j, the expert e_k associated to it a relative importance $w_j^k \in [0,1]$ such that $\sum_{j=1}^{n} w_j^k = 1$. Also, e_k defines the performance p_{ij}^k for each alternative i and for each criterion j by means of a fuzzy number. In fuzzy MCDM, the principal difference with the classical case is actually the fact that performance factors are *fuzzy numbers* defined by means of triangular membership functions $\texttt{triangular}(a,b,c)$, which are intended to be an approximation of the number b.

For instance, if there are 2 experts, 2 alternatives and 2 criteria, we may have the following decision matrix:

e_1	c_1	c_2
a_1	$\texttt{triangular}(0.6, 0.7, 0.8)$	$\texttt{triangular}(0.9, 0.95, 1)$
a_2	$\texttt{triangular}(0.6, 0.7, 0.8)$	$\texttt{triangular}(0.4, 0.5, 0.6)$
e_2	c_1	c_2
a_1	$\texttt{triangular}(0.55, 0.6, 7)$	$\texttt{triangular}(0.4, 0.45, 0.5)$
a_2	$\texttt{triangular}(0.35, 0.4, 0.45)$	$\texttt{triangular}(0.5, 0.55, 0.6)$

For this decision matrix, we may have the following weights w_j^k:

	c_1	c_2
e_1	0.48	0.52
e_2	0.52	0.48

There are many alternative methods to compute the final ranking values from the decision matrix. We will use the *Weighted Sum Method* (WSM), which is among the simplest methods in MCDM, but has the advantage to be easy embedded within fuzzy DLs. Formally, $A_i^k = \sum_{j=1}^{m} p_{ij}^k w_j^k$ is the the final ranking value of the alternative a_i according to the expert k.

The final ranking value of the alternative a_i is obtained as an average of the values obtained for every expert: $A_i = \sum_{k=1}^{p} A_i^k$.

The *ranking of the alternatives* is obtained by ordering the alternatives in descending order with respect to the final ranking value and the optimal alternative $a^\star$ is the one that maximizes the final ranking value, i.e., $a^\star = \arg\max_{a_i} A_i$.

Let us show now how to encode the previous knowledge. Every triangular membership function in the decision matrix is represented using a datatype with an annotation property indicating the parameters of the triangular membership function. For every performance p_{ij}^k we have a defined datatype a-ijk. For instance, the datatype a-211 contains the parameters of the triangular function which defines the performance for the alternative 2, criterion 1, and expert 1:

```
<fuzzyOwl2 fuzzyType="datatype">
  <Datatype type="triangular" a="0.6" b="0.7" c="0.8" />
</fuzzyOwl2>
```

For each alternative a_i, for each criterion c_j, and for each expert e_k, we define a concept Performance-ijk establishing the relation with the corresponding cell of the decision matrix. For instance, Performance-211 is defined as:

$$\text{Performance-211} = \exists \text{hasScore.a-211}$$

For each alternative a_i, and for each expert e_k, we define a concept LocalValue-ik, annotated as a weighted sum concept. For instance, LocalValue-11 is annotated as follows:

```
<fuzzyOwl2 fuzzyType="concept">
  <Concept type="weightedSum">
    <Concept type="weighted" value="0.48" base="Performance-111" />
    <Concept type="weighted" value="0.52" base="Performance-121" />
  </Concept>
</fuzzyOwl2>
```

For each alternative a_i, we define a concept GlobalValue-i, annotated as a weighted sum concept. For instance, GlobalValue-1 is annotated as follows:

```
<fuzzyOwl2 fuzzyType="concept">
  <Concept type="weightedSum">
    <Concept type="weighted" value="0.5" base="LocalValoration11" />
    <Concept type="weighted" value="0.5" base="LocalValoration12" />
  </Concept>
</fuzzyOwl2>
```

Finally, the best one is the alternative a_i maximizing the satisfiability degree of the fuzzy concept GlobalValue-i. Following our example, the satisfiability degree of GlobalValue-1 is 0.26, and the satisfiability degree of GlobalValue-2 is 0.32. Consequently, the optimal alternative is a_2.

6. Discussion

This section discusses the implementation status of our approach and compares it with the related work.

6.1. Implementation

This representation of fuzzy ontologies suggests a methodology for fuzzy ontology development. First, we can build the *core part* of the ontology by using any ontology editor supporting OWL 2, such as *Protégé* 4.1[5] [13, 18]. This allows to reason with this part using standard ontology reasoners. Then, we add the *fuzzy part* of the ontology by using annotation properties. This can also be done directly with an OWL 2 ontology editor.

Once the fuzzy ontology has been created, it has to be translated into the language supported by some fuzzy DL reasoner, so that we can reason with it. For this purpose, we have developed a template code for a parser translating from OWL 2 with annotations of type `fuzzyLabel` into the language supported by some fuzzy DL reasoner.

This general parser can be adapted to any particular fuzzy DL reasoner. As illustrative purposes, we have adapted it to the languages supported by the fuzzy DL reasoners `fuzzyDL`[6] [5] and `DeLorean`[7] [2]. The template and the parsers can be freely obtained from the web pages of `fuzzyDL` and `DeLorean`. It is important to point out that similar parsers for other fuzzy DL reasoners can be obtained without difficulties. These three parsers are publicly available on the web[8].

The parsers are based on *OWL API 3*[9] [12]. OWL API 3 is a high level Application Programming Interface for working with OWL 2 ontologies. It is becoming a de-facto standard and many SW tools already support it. OWL API allows to iterate over the elements of the ontology in a transparent way.

[5] `http://protege.stanford.edu/`
[6] `http://www.straccia.info/software/fuzzyDL/fuzzyDL.html`
[7] `http://webdiis.unizar.es/~fbobillo/delorean`
[8] `http://www.straccia.info/software/FuzzyOWL/`
[9] `http://owlapi.sourceforge.net`

Whenever an element is supported by the fuzzy DL reasoner, it is mapped into its internal representation of a fuzzy ontology. The output of the process is a fuzzy ontology, which can be printed in the standard output or saved in a text file.

Table 3: Fragments of fuzzy OWL 2 supported by the fuzzy DL reasoners `fuzzyDL` and `DeLorean`.

Concept	`fuzzyDL`	`DeLorean`
(C1)	Yes	Yes
(C2)	Yes	Yes
(C3)	Yes	Yes
(C4)	Yes	Yes
(C5)	Yes	Yes
(C6)	Yes	Yes
(C7)	Yes	Yes
(C8)	Yes	Yes
(C9)	Yes	Yes
(C10)	Yes	Yes
(C11)	No	Yes
(C12)	No	Yes
(C13)	No	Yes
(C14)	No	Yes
(C15)	No	Yes
(C16)	Yes	Yes
(C17)	Yes	Yes
(C18)	Yes	No
(C19)	Yes	No
Role	`fuzzyDL`	`DeLorean`
(R1)	Yes	Yes
(R2)	Yes	Yes
(R3)	No	Yes
(R4)	No	No
(R5)	Yes	Yes

Axiom	`fuzzyDL`	`DeLorean`
(A1)	Yes	Yes
(A2)	Yes	Yes
(A3)	No	Yes
(A4)	Yes	Yes
(A5)	Yes	Yes
(A6)	No	Yes
(A7)	No	Yes
(A8)	Yes	Yes
(A9)	Yes	Yes
(A10)	Yes	Yes
(A11)	Yes	Yes
(A12)	Partial	Yes
(A13)	Yes	Yes
(A14)	Yes	Yes
(A15)	Yes	Yes
(A16)	Yes	Yes
(A17)	Yes	Yes
(A18)	Yes	Yes
(A19)	Yes	Yes
(A20)	No	Yes
(A21)	No	Yes
(A22)	Yes	Yes
(A23)	No	Yes
(A24)	Yes	Yes
(A25)	No	Yes

A full reasoning algorithm for the logic presented in Section 3 is not known yet. Consequently, the parsers only cover the fragments of fuzzy OWL 2 currently supported by these reasoners. Table 3 summarizes the fragments of fuzzy OWL 2 supported by `fuzzyDL` and `DeLorean`[10]. Such table should not be intended as a comparison of the two reasoners. Even if `fuzzyDL` is based of fuzzy $\mathcal{SHIF}(\mathbf{D})$ instead of fuzzy $\mathcal{SROIQ}(\mathbf{D})$, there are many features that are not available in other fuzzy DL reasoner.

6.2. Related work

This is, to the best of our knowledge, the first effort towards fuzzy ontology representation using OWL 2.

Naïve fuzzy extensions of ontology languages have been presented, more precisely OWL [10, 22] and OWL 2 [21]. These languages are obviously not complaint with OWL 2 and current ontology editors, as it happens under out approach. Furthermore, they are not expressive enough since they only allow

[10]We say that `fuzzyDL` partially supports axioms (A12) because they are restricted to the case $m = 1$.

a fuzzy ABox. That is, they are restricted to a subset of our case 5, only for axioms (A1)–(A3).

A similar work provides an OWL ontology for fuzzy ontology representation [7]. There, annotation properties are not used, but concepts, roles and axioms are represented as individuals. For instance, Example 4 would be represented using the following axioms (in abstract syntax):

```
(ClassAssertion paul Individual)
(ClassAssertion tall Concept)
(ClassAssertion ax1 ConceptAssertion)
(ObjectPropertyAssertion ax1 isComposedOfAbstractIndividual)
(ObjectPropertyAssertion ax1 isComposedOfAbstractConcept)
```

However, this representation has many problems:

- Representing concepts, roles and axioms as individuals causes (meta)logical problems.
- Instead of reusing current ontology editors, the method requires a completely different and user-unfriendly way of modelling, e.g., a concept conjunction is not represented using `intersectionOf`, but using a specific encoding using a individual (representing the concept) related with two individuals (each of them representing one of the conjuncts).
- Last but no least, it is not an efficient representation, since the ontology grows exponentially with the size of the ontology.

A closer approach to ours is [14], which also uses annotation properties to add probabilistic constraints, but it is restricted to a subset of our case 5, axioms (A1) and (A8).

A pattern for uncertainty representation in ontologies has also been presented in [28]. However, it is restricted to a subset of our case 5, only for axioms (A1). Furthermore, it relies in OWL Full, thus not making possible to reason with the ontology.

Our approach should not be confused with a series of works that describe, given a fuzzy ontology, how to obtain an equivalent OWL 2 ontology (see for example [3, 4, 6, 21, 24]). In these works it is possible to reason using a crisp DL reasoner instead of a fuzzy DL reasoner, which is not our case. However, the advantage of our approach is that we provide a specific format to represent fuzzy ontologies which can be easily managed by current OWL editors and understood by humans.

The W3C Uncertainty Reasoning for the World Wide Web Incubator Group (URW3-XG) defined an ontology of uncertainty, a vocabulary which can be used to annotate different pieces of information with different types of uncertainty (e.g. vagueness, randomness or incompleteness), the nature of the uncertainty, etc. [29]. But unlike our approach, it can only be used to identify some kind of uncertainty, and not to represent and manage uncertain pieces of information.

Finally, we explain the main differences with a previous version of our work [8].

- In the previous version, there are some concept constructors that have several versions depending on the fuzzy logic considered. For instance, we had $\sqcap_G$ and $\sqcap_{\mathrm{L}}$ denoting Gödel and Łukasiewicz t-norm, respectively. This has the advantage that the user is free to combine connectives from different fuzzy connectives. However, this also has some problems. Firstly, from a practical point of view, such combinations are not clear yet from a reasoning point of view. Secondly, since OWL 2 does not make possible to annotate concept expressions, this would require to create a new named entity every time these constructors are used, which is problematic from a modelling point of view. For instance, given a concept $C_1 \sqcap_G C_2$ would require to create a new concept $D = C_1 \sqcap C_2$ and to annotate it with the semantics of the fuzzy logic.

- In the previous version, there also some axioms which have several versions depending on the fuzzy logic considered, but we do not allow them either for the sake of coherence.

- As a consequence of the previous differences, now we allow to annotate ontologies, in order to specify the fuzzy logic considered in the semantics of all the elements of the ontology.

- In the previous version, we used annotation properties of type `rdfs:comment`. Obviously, there was not a clear separation between real comments and fuzzy information. This has been solved by using a new annotation property `fuzzyLabel`.

- In the current version, we are restricted to Łukasiewicz and Zadeh fuzzy logics, which are supported by `fuzzyDL` or `DeLorean`. However, it is trivial to extend the syntax to cover alternative fuzzy logics, such as Gödel or Product.

7. Conclusions and Future Work

In this article we have dealt with the problem of fuzzy ontology representation. Instead of proposing a fuzzy extension of an ontology language as a candidate to become a standard for fuzzy ontologies, which is not foreseeable in the next years, we have proposed a framework represent fuzzy ontologies using current languages and resources.

To begin with, we have claimed that the current fuzzy extensions are not expressive enough, and have identified the syntactical differences that a fuzzy ontology language has to cope with, grouping them into 5 different cases. Our work consider a very general fuzzy extension of the DL $\mathcal{SROIQ}(\mathbf{D})$, which is the logical formalism of OWL 2. In fact, our logic is not restricted to a simple fuzzy ABox, but there are many differences with respect to the case, such as fuzzy datatypes, fuzzy modifiers or weighted sum concepts. However, our approach is extensible and can easily be augmented to support, e.g., alternative fuzzy logics, modifier functions and fuzzy datatypes.

Then, we have provided a representation using the current standard language OWL 2, by using annotation properties. A similar approach cannot be represented in OWL DL as it does not support rich enough annotation capabilities. This way, we can use OWL 2 editors to develop fuzzy ontologies. Furthermore, non-fuzzy reasoners applied over such a fuzzy OWL ontology can discard the fuzzy part, i.e., the annotations, producing the same results as if they would not exist.

This work suggests a methodology for fuzzy ontology development. First, we can build the *core part* of the ontology by using any ontology editor supporting OWL 2. This allows to reason with this part using standard ontology reasoners. Then, we add the *fuzzy part* of the ontology by using annotation properties. This can also be done directly with an OWL 2 ontology editor, even if some sort of user assistance would be highly appreciated.

In this regard, we have also developed some parsers translating from OWL 2 with annotations of type `fuzzyLabel` into the languages supported by some fuzzy DL reasoners. Firstly, we develop a general parser that can be adapted to any fuzzy DL reasoner. Then, as illustrative purposes, we adapted it to the languages supported by the fuzzy DL reasoners `fuzzyDL` and `DeLorean`. Similar parsers for other fuzzy DL reasoners could be easily obtained.

We are currently developing a graphical interface (a Protégé plug-in) to make the encoding of annotation properties transparent to the user. In future work, we would like to develop similar parsers for other fuzzy DL reasoners, such as `Fire`.

Acknowledgement

F. Bobillo has been partially funded by the Spanish Ministry of Science and Technology (project TIN2009-14538-C02-01) and Ministry of Education (program José Castillejo, grant JC2009-00337).

References

[1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[2] F. Bobillo, M. Delgado, and J. Gómez-Romero. DeLorean: A reasoner for fuzzy OWL 1.1. In *Proceedings of the 4th International Workshop on Uncertainty Reasoning for the Semantic Web (URSW 2008)*, volume 423 of CEUR Workshop Proceedings, 10 2008.

[3] F. Bobillo, M. Delgado, and J. Gómez-Romero. Crisp representations and reasoning for fuzzy ontologies. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 17(4):501–530, 2009.

[4] F. Bobillo, M. Delgado, J. Gómez-Romero, and U. Straccia. Fuzzy Description Logics under Gödel semantics. *International Journal of Approximate Reasoning*, 50(3):494–514, 2009.

[5] F. Bobillo and U. Straccia. fuzzyDL: An expressive fuzzy Description Logic reasoner. In *Proceedings of the 17th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2008)*, pages 923–930. IEEE Computer Society, 6 2008.

[6] F. Bobillo and U. Straccia. Towards a crisp representation of fuzzy Description Logics under Łukasiewicz semantics. In *Proceedings of the 17th International Symposium on Methodologies for Intelligent Systems (ISMIS 2008)*, volume 4994 of *Lecture Notes in Computer Science*, pages 309–318. Springer-Verlag, 5 2008.

[7] F. Bobillo and U. Straccia. An OWL ontology for fuzzy OWL 2. In J. R. et al., editor, *Proceedings of the 18th International Symposium on Methodologies for Intelligent Systems (ISMIS 2009)*, volume 5722 of *Lecture Notes in Computer Science*, pages 151–160. Springer-Verlag, 9 2009.

[8] F. Bobillo and U. Straccia. Representing fuzzy ontologies in OWL 2. In *Proceedings of the 19th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2010)*, pages 2695–2700. IEEE Press, July 2010.

[9] B. Cuenca-Grau, I. Horrocks, B. Motik, B. Parsia, P. F. Patel-Schneider, and U. Sattler. OWL 2: The next step for OWL. *Journal of Web Semantics*, 6(4):309–322, 2008.

[10] M. Gao and C. Liu. Extending OWL by fuzzy Description Logic. In *Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2005)*, pages 562–567. IEEE Computer Society, 11 2005.

[11] P. Hájek. *Metamathematics of Fuzzy Logic*. Kluwer, 1998.

[12] M. Horridge and S. Bechhofer. The OWL API: A Java API for working with owl 2 ontologies. In *Proceedings of the 6th International Workshop on OWL: Experiences and Directions (OWLED 2009)*, volume 529 of CEUR Workshop Proceedings, 10 2009.

[13] M. Horridge, D. Tsarkov, and T. Redmond. Supporting early adoption of OWL 1.1 with Protege-OWL and FaCT++. In *Proceedings of the 2nd International Workshop on OWL: Experience and Directions (OWLED 2006)*, volume 216 of CEUR Workshop Proceedings, 11 2006.

[14] P. Klinov and B. Parsia. Optimization and evaluation of reasoning in probabilistic description logic: Towards a systematic approach. In *Proceedings of the 7th International Semantic Web Conference (ISWC 2008)*, volume 5318 of *Lecture Notes in Computer Science*, pages 213–228. Springer-Verlag, 10 2008.

[15] T. Lukasiewicz and U. Straccia. Managing uncertainty and vagueness in Description Logics for the semantic web. *Journal of Web Semantics*, 6(4):291–308, 2008.

[16] P. S. Mostert and A. L. Shields. On the structure of semigroups on a compact manifold with boundary. *Annals of Mathematics*, 65(1):117–143, 1957.

[17] B. Motik, P. F. Patel-Schneider, and B. P. (editors). OWL 2 Web Ontology Language structural specification and functional-style syntax, 2009. [Online] Available: `http://www.w3.org/TR/owl2-syntax/`.

[18] N. F. Noy, M. Sintek, S. Decker, M. Crubezy, R. W. Fergerson, and M. A. Musen. Creating Semantic Web contents with Protégé-2000. *IEEE Intelligent Systems*, 16(2):60–71, 2001.

[19] E. Sanchez, editor. *Fuzzy Logic and the Semantic Web*, volume 1 of *Capturing Intelligence*. Elsevier Science, 2006.

[20] G. Stoilos, N. Simou, G. Stamou, and S. Kollias. Uncertainty and the semantic web. *IEEE Intelligent Systems*, 21(5):84–87, 2006.

[21] G. Stoilos and G. Stamou. Extending fuzzy Description Logics for the semantic web. In *Proceedings of the 3rd International Workshop on OWL: Experiences and Directions (OWLED 2007)*, volume 258 of CEUR Workshop Proceedings, 6 2007.

[22] G. Stoilos, G. Stamou, and J. Z. Pan. Fuzzy extensions of OWL: Logical properties and reduction to fuzzy Description Logics. *International Journal of Approximate Reasoning*, 51:656–679, 2010.

[23] U. Straccia. Reasoning within fuzzy Description Logics. *Journal of Artificial Intelligence Research*, 14:137–166, 2001.

[24] U. Straccia. Transforming fuzzy Description Logics into classical description logics. In *Proceedings of the 9th European Conference on Logics in Artificial Intelligence (JELIA 2004)*, volume 3229 of *Lecture Notes in Computer Science*, pages 385–399. Springer-Verlag, 9 2004.

[25] U. Straccia. Description logics with fuzzy concrete domains. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI 2005)*. AUAI Press, 7 2005.

[26] U. Straccia. A fuzzy Description Logic for the semantic web. In E. Sanchez, editor, *Fuzzy Logic and the Semantic Web*, volume 1 of *Capturing Intelligence*, pages 73–90. Elsevier Science, 2006.

[27] U. Straccia. Multi-criteria decision making in fuzzy Description Logics: A first step. In *Proceedings of the 13th International Conference on Knowledge-Based & Intelligent Information & Engineering Systems (KES*

2009), volume 5711 of *Lecture Notes in Artificial Intelligence*, pages 79–87. Springer-Verlag, 2009.

[28] M. Vacura, V. Svátek, and P. Smrž. A pattern-based framework for representation of uncertainty in ontologies. In *Proceedings of the 11th International Conference on Text, Speech, and Dialogue (TSD 2008)*, volume 5246 of *Lecture Notes in Computer Science*, pages 227–234. Springer-Verlag, 9 2008.

[29] W3C Incubator Group on Uncertainty Reasoning for the World Wide Web Final Report: `http://www.w3.org/2005/Incubator/urw3/XGR-urw3`, 2008.

[30] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

A. From Fuzzy DLs to Fuzzy OWL

In this article we have used fuzzy DLs as the original language to express fuzzy ontologies. As already claimed throughout this article, our objective is not provide a new fuzzy ontology language, such as fuzzy OWL 2. However, for the sake of completeness, we find useful to include as an appendix, a short note about the relation between DLs and OWL 2.

An OWL 2 ontology contains descriptions of *classes* (or concepts in DL terminology), *properties* (roles in DL terminology) and *individuals*. There are two types of properties: *object properties* (abstract roles) and *datatype properties* (concrete roles). Table 4 includes the classes and properties constructors of OWL 2, together with their correspondences in $\mathcal{SROIQ}(\mathbf{D})$.

There are two additional types of properties which do not have a counterpart in the DL, namely *annotation properties* (`owl:AnnotationProperty`) and *ontology properties* (`owl:OntologyProperty`), but they just include some meta-properties of the ontology.

An OWL 2 document consists of optional ontology headers plus any number of axioms: facts about individuals, class axioms and property axioms, which according to the DL terminology correspond to the ABox, TBox and RBox, respectively. Ontology headers are used for meta-information, ontology import and relationships. Table 5 shows the OWL 2 axioms and their equivalences in $\mathcal{SROIQ}(\mathbf{D})$.

Table 4: Class and property constructors in OWL 2

OWL 2 abstract syntax	**DL syntax**
Class (A)	A
Class (owl:Thing)	$\top$
Class (owl:Nothing)	$\bot$
ObjectIntersectionOf (C, D)	$C \sqcap D$
ObjectUnionOf (C, D)	$C \sqcup D$
ObjectComplementOf (C)	$\neg C$
ObjectAllValuesFrom (R, C)	$\forall R.C$
ObjectSomeValuesFrom (R, C)	$\exists R.C$
ObjectHasValue (R, o)	$\exists R.\{o\}$
DataAllValuesFrom (T, d)	$\forall T.\mathbf{d}$
DataSomeValuesFrom (T, d)	$\exists T.\mathbf{d}$
DataHasValue (T, v)	$\exists T.\{v\}$
ObjectOneOf ($o_1, \ldots, o_m$)	$\{o_1\} \sqcup \{o_2\} \sqcup \{o_m\}$
ObjectMinCardinality (n, S, C)	$(\geq n\ S.C)$
ObjectMaxCardinality (n, S, C)	$(\leq n\ S.C)$
ObjectExactCardinality (n, S, C)	$(\geq n\ S.C) \sqcap (\leq n\ S.C)$
ObjectMinCardinality (n, S)	$(\geq n\ S.\top)$
ObjectMaxCardinality (n, S)	$(\leq n\ S.\top)$
ObjectExactCardinality (n, S)	$(\geq n\ S.\top) \sqcap (\leq n\ S.\top)$
DataMinCardinality (n, T, d)	$(\geq n\ T.\mathbf{d})$
DataMaxCardinality (n, T, d)	$(\leq n\ T.\mathbf{d})$
DataExactCardinality (n, T, d)	$(\geq n\ T.\mathbf{d}) \sqcap (\leq n\ T.\mathbf{d})$
DataMinCardinality (n, T)	$(\geq n\ T.\top)$
DataMaxCardinality (n, T)	$(\leq n\ T.\top)$
DataExactCardinality (n, T)	$(\geq n\ T.\top) \sqcap (\leq n\ T.\top)$
ObjectExistsSelf (S)	$\exists S.\texttt{Self}$
ObjectProperty (R_A)	R_A
TopObjectProperty	U
BottomObjectProperty	$\neg U$
DatatypeProperty (T)	T
TopDataProperty	U_D
BottomDataProperty	$\neg U_D$

Table 5: Axioms in OWL 2

OWL 2 abstract syntax	**DL syntax**
ClassAssertion (a, C)	$a\colon C$
ObjectPropertyAssertion $(R, a, b))$	$(a, b)\colon R$
NegativeObjectPropertyAssertion (R, a, b)	$(a, b)\colon \neg R$
DataPropertyAssertion $(T, a, v))$	$(a, v)\colon T$
NegativeDataPropertyAssertion (T, a, v)	$(a, v)\colon \neg T$
SameIndividual $(a_1, \ldots, a_m)$	$a_i = a_j, 1 \leq i < j \leq m$
DifferentIndividuals $(a_1, \ldots, a_m)$	$a_i \neq a_j, 1 \leq i < j \leq m$
SubClassOf (C_1, C_2)	$C_1 \sqsubseteq C_2$
EquivalentClasses $(C_1, \ldots, C_m)$	$C_1 \equiv \cdots \equiv C_m$
DisjointClasses $(C_1, \ldots, C_m)$	$\mathtt{dis}(C_1, \ldots, C_m)$
DisjointUnion $(C, C_1, \ldots, C_m)$	$\mathtt{disUnion}(C_1, \ldots, C_m)$
SubObjectPropertyOf (subObjectPropertyChain $(R_1, \ldots, R_m)$ R)	$R_1 \ldots R_m \sqsubseteq R$
SubObjectPropertyOf $(R_1 R_2)$	$R_1 \sqsubseteq R_2$
SubDataPropertyOf (T_1, T_2)	$T_1 \sqsubseteq T_2$
EquivalentObjectProperties $(R_1, \ldots, R_m)$	$R_1 \equiv \cdots \equiv R_m$
EquivalentDataProperties $(T_1, \ldots, T_m)$	$T_1 \equiv \cdots \equiv T_m$
ObjectPropertyDomain (R, C)	$\mathtt{domain}(R, C)$
ObjectPropertyRange (R, C)	$\mathtt{range}(R, C)$
DataPropertyDomain (T, d)	$\mathtt{domain}(T, \mathbf{d})$
DataPropertyRange (T, d)	$\mathtt{range}(T, \mathbf{d})$
InverseObjectProperties (R_1, R_2)	$R_1 \equiv R_2^-$
FunctionalObjectProperty (S)	$\mathtt{func}(S)$
FunctionalDataProperty (T)	$\mathtt{func}(T)$
InverseFunctionalObjectProperty (S)	$\mathtt{func}(S^-)$
TransitiveObjectProperty (R)	$\mathtt{trans}(R)$
DisjointObjectProperties (S_1, S_2)	$\mathtt{dis}(S_1, S_2)$
DisjointDataProperties (T_1, T_2)	$\mathtt{dis}(T_1, T_2)$
ReflexiveObjectProperty (R)	$\mathtt{ref}(R)$
IrreflexiveObjectProperty (S)	$\mathtt{irr}(S)$
SymmetricObjectProperty (R)	$\mathtt{sym}(R)$
AsymmetricObjectProperty (S)	$\mathtt{asy}(S)$