International Journal of Approximate Reasoning 52 (2011) 1073–1094

Contents lists available at SciVerse ScienceDirect

International Journal of Approximate Reasoning

journal homepage: www.elsevier.com/locate/ijar

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

ARTICLE INFO

Article history:
Received 17 September 2010
Revised 6 May 2011
Accepted 16 May 2011
Available online 23 May 2011

Keywords:
Fuzzy OWL 2
Fuzzy ontologies
Fuzzy languages for the Semantic Web
Fuzzy description logics

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 some prototypical implementations: a plug-in to edit fuzzy ontologies using OWL 2 annotations and some parsers that translate fuzzy ontologies represented using our methodology into the languages supported by some reasoners.

1. Introduction

Today, there is a growing interest in the development of knowledge representation formalisms able to deal with uncertainty, 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 [57].

Since fuzzy set theory and fuzzy logic [60] are suitable formalisms to handle these types of knowledge. It is not surprising that fuzzy ontologies are useful in several applications, ranging from information retrieval [14,28,49], image interpretation [17,18,26], the Semantic Web and the Internet [15,45,48], among many others [12,13,29–32,46,47,56].

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 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 [16,59], is equivalent to the DL $\mathcal{SROIQ}(\mathbf{D})$.

Several fuzzy extensions of DLs can be found in the literature. For a good survey on the topic, we refer the reader to [33]; some examples of recent work in the field include [8,20,21,34–36,38,52]. In addition to the theoretical research, some fuzzy DL reasoners have been implemented, such as fuzzyDL [7], DeLorean [4] and Fire [50]. 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 a standard one of the fuzzy extensions of the languages OWL and OWL 2 that have been proposed, such as [19,51,52]. However, we do not expect a fuzzy OWL extension to become a W3C proposed standard

☆ 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).

* Corresponding author.
E-mail addresses: fbobillo@unizar.es (F. Bobillo), straccia@isti.cnr.it (U. Straccia).

0888-613X/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.ijar.2011.05.003

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	$\alpha \cdot \beta$	$\alpha + \beta - \alpha \cdot \beta$	$\begin{cases} 1, & \alpha = 0 \\ 0, & \alpha > 0 \end{cases}$	$\begin{cases} 1, & \alpha \leq \beta \\ \beta/\alpha, & \alpha > \beta \end{cases}$

in the near future. Furthermore, we argue that current fuzzy extensions are not expressive enough, as they only provide syntactic modifications in some of the axioms of the ontology (in the ABox).

In this work, we propose to use OWL 2 itself to represent fuzzy ontologies. More precisely, we identify the syntactic differences that a fuzzy ontology language has to cope with, and show how to encode them using OWL 2 annotation properties. 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 almost the same results as if it would not exist (however, as we will see in Section 3, our methodology may need to introduce new concepts or roles that cannot be directly discarded).

In order to support our methodology for fuzzy ontology representation, we have implemented a Protégé plug-in to edit fuzzy ontologies and some parsers that translate fuzzy ontologies represented using our methodology into the languages supported by some fuzzy DL reasoners.

The remainder of this paper is organized as follows. Section 2 includes some preliminaries that will be used in the rest of the paper. More precisely, Section 2.1 is dedicated to fuzzy logic, Section 2.2 to our fuzzy extension of OWL 2, and Section 2.3 to the different syntaxes of OWL 2. Section 3 presents the main contribution of our work, showing how to encode it fuzzy OWL 2 ontologies using OWL 2. Section 4 illustrates the methodology with some application problems. Section 5 discusses the implementation status of our approach. In particular, Section 5.1 describes a plug-in to edit fuzzy ontologies, Section 5.2 describes some parsers to export fuzzy ontologies, and Section 5.3 evaluates its practical behaviour with some experiments. Next, Section 6 compares our approach with the related work. Finally, Section 7 sets out some conclusions and ideas for future research.

2. Preliminaries

This section recalls some background knowledge on fuzzy logic (Section 2.1), the language fuzzy OWL 2 (Section 2.2), and the different syntaxes of OWL 2 (Section 2.3).

2.1. Fuzzy logic

Fuzzy set theory and fuzzy logic were proposed by Zadeh [60] 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 type of propositions, called *fuzzy propositions*. Each fuzzy proposition may have a *degree of truth* in [0, 1], denoting the compatibility of the fuzzy proposition with a given state of facts. For example, the truth of the proposition stating than a given tomato is a ripe tomato is clearly a matter of degree.

In this article we will consider *fuzzy formulae* (or fuzzy axioms) of the form $\phi \geq \alpha$ or $\phi \leq \beta$, where ϕ is a fuzzy proposition and $\alpha, \beta \in [0, 1]$ [22]. This imposes that the degree of truth of ϕ is *at least* α (resp. *at most* β). For example, x is a ripe tomato ≥ 0.9 says that we have a rather ripe tomato (the degree of truth of x being a ripe tomato 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 [22].

There are three main fuzzy logics: Łukasiewicz, Gödel, and Product. The importance of these three fuzzy logics is due to the fact that any continuous t-norm can be obtained as a combination of Łukasiewicz, Gödel, and Product t-norm [39]. 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.

A (binary) *fuzzy relation* R over two countable classical sets X and Y is a function $R: X \times Y \rightarrow [0, 1]$. Again, all crisp operations over relations (e.g., reflexivity, symmetry, or transitivity) are extended to the fuzzy case.

2.2. Fuzzy OWL 2

In this section we describe the syntax of the fuzzy extension of OWL 2 that we will consider in the rest of the paper. Fuzzy extensions of OWL 2 have a very close connection to the fuzzy DL $\mathcal{SROIQ}(\mathbf{D})$ (see for instance [52]). In this section, we will use the more concrete and less cumbersome DL notation instead of the OWL 2 one.

In this paper we will focus on syntactic issues, but the interested reader may find in the literature the semantics, logical properties and reasoning algorithms for Zadeh fuzzy logic [5], Gödel fuzzy logic [6], and Łukasiewicz fuzzy logic [11].

Alphabet. Fuzzy OWL 2 assumes three alphabets of symbols, for *fuzzy concepts*, *fuzzy roles* and *individuals*. In fuzzy OWL 2, fuzzy concepts denote fuzzy sets of individuals and fuzzy roles denote fuzzy binary relations.

Notation. To begin with, we will 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,
- $\rhd \in \{\geq, >\}$, $\bowtie \in \{\geq, >, \leq, <\}$,
- $\alpha \in [0, 1]$.

Next, we will introduce two important elements of our logic: fuzzy modifiers and fuzzy concrete domains which have been presented in [54].

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

$$mod \to \quad \texttt{linear}(c) \quad | \quad \text{(M1)}$$
$$\texttt{triangular}(a, b, c) \quad \text{(M2)}$$

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

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

Fuzzy concrete domains. A *fuzzy concrete domain* (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 (Fig. 1a), the *triangular* (Fig. 1b), the *left-shoulder* function (Fig. 1c) and the *right-shoulder* function (Fig. 1d).

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

Formally:

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

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

[1] Intuitively, simple roles cannot take part in cyclic role inclusion axioms (see [5] for a formal definition).

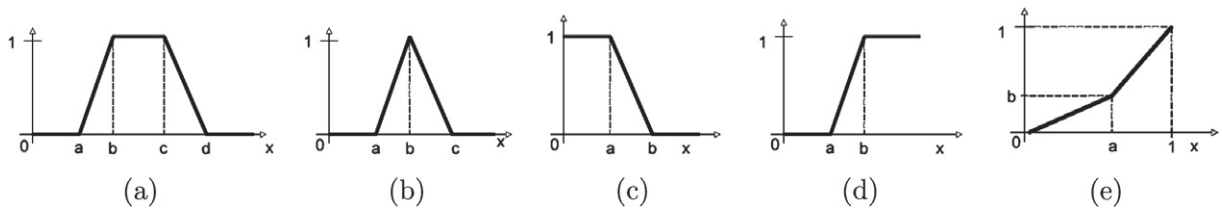


Fig. 1. (a) Trapezoidal function; (b) triangular function; (c) left-shoulder function; (d) right-shoulder function; and (e) linear function.

Table 2
Syntax of fuzzy OWL 2.

Concept	Syntax	Axiom	Syntax
(C1)	A	(A1)	$\langle a : C \bowtie \alpha \rangle$
(C2)	$\top$	(A2)	$\langle (a, b) : R \bowtie \alpha \rangle$
(C3)	$\bot$	(A3)	$\langle (a, b) : \neg R \bowtie \alpha \rangle$
(C4)	$C \sqcap D$	(A4)	$\langle (a, v) : T \bowtie \alpha \rangle$
(C5)	$C \sqcup D$	(A5)	$\langle (a, v) : \neg T \bowtie \alpha \rangle$
(C6)	$\neg C$	(A6)	$\langle a \neq b \rangle$
(C7)	$\forall R.C$	(A7)	$\langle a = b \rangle$
(C8)	$\exists R.C$	(A8)	$\langle C \sqsubseteq D \triangleright \alpha \rangle$
(C9)	$\forall T.\mathbf{d}$	(A9)	$C_1 \equiv \ldots C_m$
(C10)	$\exists T.\mathbf{d}$	(A10)	$\mathtt{dis}(C_1, \ldots, C_m)$
(C11)	$\{\alpha/a\}$	(A11)	$\mathtt{disUnion}(C_1, \ldots, C_m)$
(C12)	$\geq m\ S.C$	(A12)	$\langle R_1 \ldots R_m \sqsubseteq R \triangleright \alpha \rangle$
(C13)	$\leq n\ S.C$	(A13)	$\langle T_1 \sqsubseteq T_2 \triangleright \alpha \rangle$
(C14)	$\geq m\ T.\mathbf{d}$	(A14)	$R_1 \equiv \ldots R_m$
(C15)	$\leq n\ T.\mathbf{d}$)	(A15)	$T_1 \equiv \ldots T_m$
(C16)	$\exists S.\mathtt{Self}$	(A16)	$\mathtt{domain}(R, C)$
(C17)	$mod(C)$	(A17)	$\mathtt{range}(R, C)$
(C18)	$\alpha \cdot C$	(A18)	$\mathtt{range}(T, \mathbf{d})$
(C19)	$(\alpha_1 \cdot C_1) + \cdots + (\alpha_k \cdot C_k)$	(A19)	$\mathtt{func}(S)$
Role	**Syntax**	(A20)	$\mathtt{func}(T)$
(R1)	R_A	(A21)	$R \equiv R^-$
(R2)	T	(A22)	$\mathtt{trans}(R)$
(R3)	R^-	(A23)	$\mathtt{dis}(S_1, \ldots, S_m)$
(R4)	U	(A24)	$\mathtt{dis}(T_1, \ldots, T_m)$
(R5)	$mod(R)$	(A25)	$\mathtt{ref}(R)$
Datatype	**Syntax**	(A26)	$\mathtt{irr}(S)$
(D1)	$\mathtt{left}(k_1, k_2, a, b)$	(A27)	$\mathtt{sym}(R)$
(D2)	$\mathtt{right}(k_1, k_2, a, b)$	(A28)	$\mathtt{asy}(S)$
(D3)	$\mathtt{triangular}(k_1, k_2, a, b, c)$		
(D4)	$\mathtt{trapezoidal}(k_1, k_2, a, b, c, d)$		
(D5)	$mod(\mathbf{d})$		

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

Concepts. The syntax of fuzzy concepts is shown in Table 2. Concept constructors (C1)–(C16) correspond to the concept constructors of OWL 2. The new concepts are modified concepts (C17), weighted concepts (C18), and weighted sum concepts (C19). In (C19), we assume that $\sum_{i=1}^{k} \alpha_i \leq 1$.

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

Roles. The syntax of fuzzy roles is shown in Table 2. Role constructors (R1)–(R4) correspond to the role constructors of OWL 2. (R5) corresponds to modified roles.

Fuzzy knowledge base. A *fuzzy Knowledge Base* (KB) or *fuzzy ontology* is a finite set 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)–(A28). All the axioms correspond to the axioms of OWL 2.

In axioms (*A*8), (*A*12), (*A*13) we argue that it does not make sense to have axioms of the forms $\langle \tau \leq \alpha \rangle$ or $\langle \tau < \alpha \rangle$ because such axioms do not have an equivalent expression in classical OWL 2.

Example 4. The fuzzy concept assertion $\langle \mathsf{paul} : \mathsf{Tall} \geq 0.5 \rangle$ 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.

In this work, we consider fuzzy OWL 2 ontologies, and we need the syntactic restrictions of simple roles to guarantee the decidability of the logic. We note that one could consider less expressive ontology languages where this restriction can be removed, such as fuzzy OWL 2 EL, which is closely related to the fuzzy DL $\mathcal{EL}++$ [37]. In this case, the same procedure to represent fuzzy ontologies described in Section 3 could still be used: just let S be any fuzzy role, and not necessarily a simple one.

2.3. OWL 2 syntaxes

In order to store and to exchange OWL 2 ontologies, concrete syntaxes are needed. For this purpose, OWL 2 provides several different syntaxes [59]. The aim of this section is to give an overview of all of them. For the sake of concrete illustration, we will show how to represent an annotated entity using each of the syntaxes.

The main syntax for OWL 2 is *RDF/XML syntax*, which defines an XML serialization for RDF triples (or RDF graphs) [2]. The basic idea is to represent the nodes and predicates of the RDF triples using XML terms. It is the only mandatory syntax, which means that it must be supported by every OWL 2 tool. Thus, it is the most appropriate syntax to improve the interoperability.

Example 5. Assume that an OWL 2 concept *className* is annotated via an annotation property *annotationProperty* with a value *annotationValue*. In RDF/XML syntax, this is represented as follows:

```
<owl:Class rdf:about="className">
   <annotationProperty>annotationValue</annotationProperty>
</owl:Class>
```

Using XML imposes several restrictions. This has motivated the emergence of alternative RDF serializations, such as *Turtle* [3], that makes reading and writing RDF triples easier.

Example 6. Example 5 is represented in Turtle syntax as follows:

```
:className rdf:type owl:Class ;
   :annotationProperty annotationValue .
```

OWL 2 has a core part or *structural specification* that determines its conceptual structure and is independent of any concrete syntax. The *functional-style syntax* (also called abstract syntax) closely corresponds to the structural specification [41]. It is a compact syntax that makes easier to see the structure of the ontologies.

Example 7. Example 5 is represented in functional-style syntax as follows:

```
AnnotationAssertion(
   annotationProperty className annotationValue
)
```

The *OWL/XML syntax* defines an XML serialization for OWL 2 ontologies, mirroring the structural specification [40]. It can be seen as a notational variant of the functional syntax, with the advantage of being easier to process using XML tools. As we will see, this is the syntax that we have chosen to represent our examples in Section 4.

Example 8. Example 5 is represented in OWL/XML syntax as follows:

```
<AnnotationAssertion>
   <AnnotationProperty IRI="#annotationProperty"/>
   <IRI>#className</IRI>
   <Literal datatypeIRI="&rdf;PlainLiteral">annotationValue</Literal>
</AnnotationAssertion>
```

Finally, the *Manchester Syntax* is specifically designed to be readable, so it is easily understood by humans [24]. It is also compact and closer to DL syntax than other syntaxes.

Example 9. Example 5 is represented in Manchester syntax as follows:

```
Class: className
   Annotations:
        annotationProperty annotationValue
```

3. Representation of fuzzy ontologies in OWL 2

In this section we will explain a methodology to represent fuzzy OWL 2 ontologies using OWL 2. The idea of our representation is to use an OWL 2 ontology and to extend 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/XML syntax [40].[2]

It is worth to note that only OWL 2 provides for annotations on ontologies, axioms, and entities [40]. This is not the case of OWL DL, which just provides for annotations on ontologies and entities.

3.1. Syntactic requirements of fuzzy ontologies

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

Case 1. Fuzzy modifiers do not have an equivalence in the non-fuzzy case: (M1), (M2).
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: (R5).
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.

3.2. 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.

3.3. Fuzzy modifiers

According to Section 2.2, the fuzzy modifiers that we want to represent have parameters a, b, c. In this case, the value of `fuzzyType` is `modifier`, and there is a tag `Modifier` with an attribute `type` (possible values `linear`, and `triangular`), and attributes `a`, `b`, `c`, depending on the type of the modifier.

Note that, differently from the case of fuzzy datatypes that we will discuss in Section 3.4, we do not need to define the values `xsd:minInclusive` and `xsd:maxInclusive` as they are assumed to be 0 and 1, respectively.

Domain of the annotation. An OWL 2 datatype 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>" />
```

`<DOUBLE>` denotes a rational number.

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 10. In order to define the fuzzy modifier Very $=$ `linear`(0.8), a datatype Very is annotated as follows:

```
<AnnotationAssertion>
   <AnnotationProperty IRI='#fuzzyLabel'/>
   <IRI>#very</IRI>
   <Literal datatypeIRI='&rdf;PlainLiteral'>
      <fuzzyOwl2 fuzzyType="modifier">
         <Modifier type="linear" c="0.8" />
```

[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.

```
      </fuzzyOwl2>
    </Literal>
</AnnotationAssertion>
```

3.4. Fuzzy datatypes

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

3.4.1. Fuzzy atomic datatypes

According to Section 2.2, 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 of the type base of the fuzzy datatype (integer `xsd:integer` or double `xsd:double`), such that:

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

`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 11. 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>
   <Datatype IRI='#YoungAge'/>
   <DataIntersectionOf>
      <DatatypeRestriction>
        <Datatype abbreviatedIRI='xsd:double'/>
        <FacetRestriction facet='&xsd;minInclusive'>
           <Literal datatypeIRI='&xsd;integer'>0</Literal>
        </FacetRestriction>
      </DatatypeRestriction>
      <DatatypeRestriction>
        <Datatype abbreviatedIRI='xsd:double'/>
        <FacetRestriction facet='&xsd;maxInclusive'>
           <Literal datatypeIRI='&xsd;integer'>200</Literal>
        </FacetRestriction>
      </DatatypeRestriction>
   </DataIntersectionOf>
</DatatypeDefinition>
```

Next, we add the annotation property as follows:

```
<AnnotationAssertion>
   <AnnotationProperty IRI='#fuzzyLabel'/>
   <IRI>#YoungAge</IRI>
   <Literal datatypeIRI='&rdf;PlainLiteral'>
     <fuzzyOwl2 fuzzyType="datatype">
        <Datatype type="leftshoulder" a="10" b="30" />
     </fuzzyOwl2>
   </Literal>
</AnnotationAssertion>
```

3.4.2. 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.

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* is defined as a fuzzy modifier.
- *base* is defined as a fuzzy datatype.
- *base* has a different name than the annotated datatype.

Example 12. Let us represent the fuzzy datatype VeryYoungAge. To begin with, we assume that the fuzzy datatype very has been created as in Example 10, and that the fuzzy datatype YoungAge has been created as in Example 11. Then, we define a new datatype VeryYoungAge and add the following annotation:

```
<AnnotationAssertion>
   <AnnotationProperty IRI="#fuzzyLabel"/>
   <IRI>#VeryYoungAge</IRI>
   <Literal datatypeIRI="&rdf;PlainLiteral">
      <fuzzyOwl2 fuzzyType="datatype">
         <Datatype type="modified" modifier="very" base="YoungAge" />
      </fuzzyOwl2>
   </Literal>
</AnnotationAssertion>
```

3.5. Fuzzy concepts

In this case, we create a new concept D and 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.

3.5.1. Fuzzy modified concepts

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

Domain of the annotation. An OWL 2 concept.

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* is defined as a fuzzy modifier.
- *base* has a different name than the annotated concept.

Example 13. Let us represent now the concept very(C). We assume that the fuzzy modifier has been created as in Example 10. To that end, we create the atomic concept VeryC and add the following annotation:

```
<AnnotationAssertion>
   <AnnotationProperty IRI='#fuzzyLabel'/>
   <IRI>#VeryC</IRI>
   <Literal datatypeIRI='&rdf;PlainLiteral'>
      <fuzzyOwl2 fuzzyType="concept">
         <Concept type="modified" modifier="very" base="C" />
      </fuzzyOwl2>
   </Literal>
</AnnotationAssertion>
```

3.5.2. 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.

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 $(0, 1]$.
- *base* has a different name than the annotated concept.

Example 14. Let us represent now the concept $(0.8\ \mathsf{C})$. We create the atomic concept $\mathsf{Weight0.8C}$ and add the following annotation:

```
<AnnotationAssertion>
   <AnnotationProperty IRI='#fuzzyLabel'/>
   <IRI>#Weight0.8C</IRI>
   <Literal datatypeIRI='&rdf;PlainLiteral'>
      <fuzzyOwl2 fuzzyType="concept">
         <Concept type="weighted" value="0.8" base="C" />
      </fuzzyOwl2>
   </Literal>
</AnnotationAssertion>
```

3.5.3. 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.

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:

- $k \geq 2$.
- $\sum_{i=1}^{k} value_k \leq 1$.
- The k base concepts have a different name than the annotated concept.

Example 15. Let us represent now the concept $(0.8\ \mathsf{A} + 0.2\ \mathsf{B})$. We create the atomic concept $\mathsf{Sum08Aplus02B}$ and add the following annotation:

```
<AnnotationAssertion>
   <AnnotationProperty IRI='#fuzzyLabel'/>
   <IRI>#Sum08Aplus02B</IRI>
   <Literal datatypeIRI='&rdf;PlainLiteral'>
      <fuzzyOwl2 fuzzyType="concept">
         <Concept type="weightedSum">
            <Concept type="weighted" value="0.8" base="A" />
            <Concept type="weighted" value="0.2" base="B" />
         </Concept>
      </fuzzyOwl2>
   </Literal>
</AnnotationAssertion>
```

3.5.4. 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.

Syntax for the annotation.

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

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

Semantical restrictions. The parsers should check the following restrictions:

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

Example 16. Let us represent now the concept $\{0.75/\mathsf{ind}\}$. We create the atomic concept ind075 and add the following annotation:

```
<AnnotationAssertion>
   <AnnotationProperty IRI='#fuzzyLabel'/>
   <IRI>#ind075</IRI>
   <Literal datatypeIRI='&rdf;PlainLiteral'>
      <fuzzyOwl2 fuzzyType="concept">
         <Concept type="nominal" value="0.75" individual="ind" />
      </fuzzyOwl2>
   </Literal>
</AnnotationAssertion>
```

3.6. Fuzzy roles

Now, we create a new role *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. Recursion is not allowed in the definitions. For the moment, we only support fuzzy modified roles.

3.6.1. Fuzzy modified roles

Here, the value of `type` is `modified`. There are also two additional attributes: `modifier` (the 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.

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* is defined as a fuzzy modifier.
- *base* has a different name than the annotated role.

Example 17. Let us represent now the abstract role very(R). We assume that the fuzzy modifier has been created as in Example 10. Then, we create the atomic object property VeryR and add the following annotation:

```
<AnnotationAssertion>
   <AnnotationProperty IRI='#fuzzyLabel'/>
   <IRI>#VeryR</IRI>
   <Literal datatypeIRI='&rdf;PlainLiteral'>
      <fuzzyOwl2 fuzzyType="role">
         <Role type="modified" modifier="very" base="R" />
      </fuzzyOwl2>
   </Literal>
</AnnotationAssertion>
```

3.7. 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.

Note that in axioms (A1)–(A5) $\langle \tau \leq \alpha \rangle$ is equivalent to $\langle \neg\tau \geq 1 - \alpha \rangle$.[3]

Domain of the annotation. An OWL 2 axiom of the following types: concept assertion,role assertion, GCI, RIA. That is, the OWL 2 versions 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 18. The fuzzy concept assertion of Example 4, $\langle \text{paul} : \text{Tall} \geq 0.5 \rangle$, is represented by annotating the concept assertion with the degree ≥ 0.5, which is done as follows:

```
<ClassAssertion>
   <Class IRI='#Tall'/>
   <NamedIndividual IRI='#paul'/>
   <Annotation>
      <AnnotationProperty IRI='#fuzzyLabel'/>
      <Literal datatypeIRI='&rdf;PlainLiteral'>
         <fuzzyOwl2 fuzzyType="axiom">
            <Degree value="0.5" />
         </fuzzyOwl2>
      </Literal>
   </Annotation>
</ClassAssertion>
```

3.8. 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=<FUZZY_LOGIC> />
</fuzzyOwl2>

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

At the moment, we only allow two fuzzy logics, Łukasiewicz and Zadeh (see Section 2.1), 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.

3.9. Non-trivial extensions

In this section, we extend our previous methodology to represent fuzzy ontologies with some additional features of fuzzy ontologies. These extensions are non-trivial and have been separated from the previous methodology because they are likely difficult to implement in practice.

3.9.1. Concepts that can be annotated with a fuzzy logic

It is possible to allow some concept constructors to have several versions depending on the fuzzy logic considered. For instance, we may have several conjunction concepts, such as $C_1 \sqcap_G C_2$ and $C_1 \sqcap_Ł C_2$ denoting Gödel conjunction and Łukasiewicz conjunction, respectively. Typically, we could specify a fuzzy logic in the concepts (C4)–(C10), (C11)–(C15). If no fuzzy logic is specified, the default value is the fuzzy logic of the ontology, represented as explained in Section 3.8.

[3] $\neg\tau$ denotes the negation of an axiom τ and is defined as follows: $\neg(a:C) = a:\neg C$, $\neg((a,b):X) = (a,b):\neg X$, $\neg((a,b):\neg X) = (a,b):X$, where $X \in \{R, T\}$.

It is important to stress that it is only possible to add annotation properties to entities (named concepts), since OWL 2 does not allow to add annotations to anonymous concept expressions. Hence, in order to add an annotation property to an anonymous concept expression, it is firstly mandatory to name it.

In order to represent one of these concepts, we create a new named concept, state that it is equivalent to the anonymous fuzzy concept, and add an annotation with a value of `fuzzyType` being `concept`, and a tag `FuzzyLogic` with an attribute `logic`, that specifies the fuzzy logic.

Domain of the annotation. An OWL 2 concept.

Syntax for the annotation.

```
<fuzzyOwl2 fuzzyType="concept">
   <FuzzyLogic logic=<FUZZY_LOGIC> />
</fuzzyOwl2>

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

Semantical restrictions. The concept is asserted to be equivalent to exactly one concept, which has one of the following types: (C4)–(C10), (C11)–(C15).

Example 19. In order to represent a fuzzy concept representing the set of tall or fat individuals, where the disjunction is interpreted using Zadeh fuzzy logic (i.e., $\mathsf{Tall} \sqcup_Z \mathsf{Fat}$, we use the new atomic concept TallOrFat as follows:

```
<EquivalentClasses>
   <Class IRI="#TallOrFat"/>
      <ObjectUnionOf>
         <Class IRI="#Tall"/>
         <Class IRI="#Fat"/>
   </ObjectUnionOf>
</EquivalentClasses>

<AnnotationAssertion>
   <AnnotationProperty IRI='#fuzzyLabel'/>
   <IRI>#TallOrFat</IRI>
   <Literal datatypeIRI='&rdf;PlainLiteral'>
      <fuzzyOwl2 fuzzyType="concept">
         <FuzzyLogic logic="zadeh" />
      </fuzzyOwl2>
   </Literal>
</AnnotationAssertion>
```

This extension has the advantage that the user can combine connectives from different fuzzy logics. However, we have several reasons to recommend not to use this feature for the moment. Firstly, from a practical point of view, such combinations are not clear yet from a reasoning point of view. Secondly, a new named entity is created every time these constructors are used. This is problematic both from a modelling and from a practical point of view, as the parsing time would increase.

3.9.2. Axioms that can be annotated with a fuzzy logic

Furthermore, it is possible to allow some axioms to have several versions depending on the fuzzy logic considered. Typically, we could specify a fuzzy logic in the axioms (A3), (A5), (A8), (A11), (A12), (A13), (A22).

Similarly as in the previous case, we add an annotation to the axiom where the value of `fuzzyType` is `axiom`, and a tag `FuzzyLogic` with and attribute `logic`, that specifies the fuzzy logic. Note that, except in the case of axioms (A11), the axiom may also have a tag that specifies a degree of truth, as shown in Section 3.7. Hence, the syntax to annotate axioms in Section 3.7 can be updated as follows.

Domain of the annotation. An OWL 2 axiom.

Syntax for the annotation.

```
<fuzzyOwl2 fuzzyType="axiom">
   <Degree value="<DOUBLE>" /> |
   <FuzzyLogic logic=<FUZZY_LOGIC> /> |
   <Degree value="<DOUBLE>" />
   <FuzzyLogic logic=<FUZZY_LOGIC> />
</fuzzyOwl2>

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

Semantical restrictions. An axiom has one of the following forms: (A3), (A5), (A8), (A11), (A12), (A13), (A22).

Example 20. Let us show how to represent the fuzzy RIA $\langle \text{isFriendOf isFriendOf} \sqsubseteq \text{isFriendOf} \geq 0.75 \rangle$, originally proposed in Example 4, using Zadeh fuzzy logic:

```
<SubObjectPropertyOf>
   <ObjectPropertyChain>
      <ObjectProperty IRI="#isFriendOf"/>
      <ObjectProperty IRI="#isFriendOf"/>
   </ObjectPropertyChain>
   <ObjectProperty IRI="#isFriendOf"/>
   <Annotation>
      <AnnotationProperty IRI="#fuzzyLabel"/>
      <Literal datatypeIRI="&rdf;PlainLiteral">
         <fuzzyOwl2 fuzzyType="concept">
            <FuzzyLogic logic="zadeh" />
            <Degree value="0.75" />
         </fuzzyOwl2>
      </Literal>
   </Annotation>
</SubObjectPropertyOf>
```

However, we propose not to allow this feature for the moment, because that seems the more coherent choice if we do not allow concept with different versions depending on the fuzzy logic, and because, as in the previous case, the combination of different axioms with semantics based on different fuzzy logics is not clear yet from a reasoning point of view.

4. Examples of fuzzy ontology representation

In this section, we will provide some examples illustrating how to use fuzzy ontologies to model the knowledge in real application problems, and how to encode fuzzy ontologies using the methodology explained in Section 3.[4] We will focus on just two applications of fuzzy ontologies: matchmaking problems and fuzzy multi-criteria decision making (MCDM) problems, but we would like to mention again that there are many more [12–15,17,18,26,28–32,45–49,56].

4.1. Matchmaking

The following example is a modified version of the one in [7]. Assume that a car seller sells a sedan car. A buyer is looking for a second hand passenger car. Both the buyer and the seller have (hard) restrictions and (soft) preferences. We have also some background knowledge about the application domain. Our aim is to find the best agreement between the buyer and the seller.

Let us show now how to represent the relevant knowledge. A concept Buy collects all the buyer's preferences together in such a way that the higher the maximal degree of satisfiability of Buy, the more the buyer is satisfied. As an example, the buyer has 5 preferences $B1$–$B5$.

$$\text{Buy} \equiv \text{BuyerRequirements} \sqcap \text{BuyerPreferences}$$

$$\text{BuyerRequirements} \equiv \text{PassengerCar} \sqcap \exists \text{hasPrice.leq26000}$$

$$\text{B1} \equiv \neg(\exists \text{hasAlarmSystem.AlarmSystem}) \sqcup \exists \text{hasPrice.ls22300-22750}$$

$$\text{B2} \equiv (\exists \text{hasInsurance.DriverInsurance}) \sqcap \exists \text{hasInsurance.}(\text{TheftInsurance} \sqcup \text{FireInsurance})$$

$$\text{B3} \equiv (\exists \text{hasAirConditioning.AirConditioning}) \sqcap \exists \text{HasExColor.}(\text{ExColorBlack} \sqcup \text{ExColorGray})$$

$$\text{B4} \equiv \exists \text{hasPrice.ls22000-24000}$$

$$\text{B5} \equiv \exists \text{hasKMWarranty.rs15000-175000}$$

Some preferences can be more important than others, so we use a weighted sum concept BuyerPreferences, and we add the following annotation property to it (see Fig. 2):

```
<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>
```

[4] The full examples are available at http://www.straccia.info/software/FuzzyOWL.

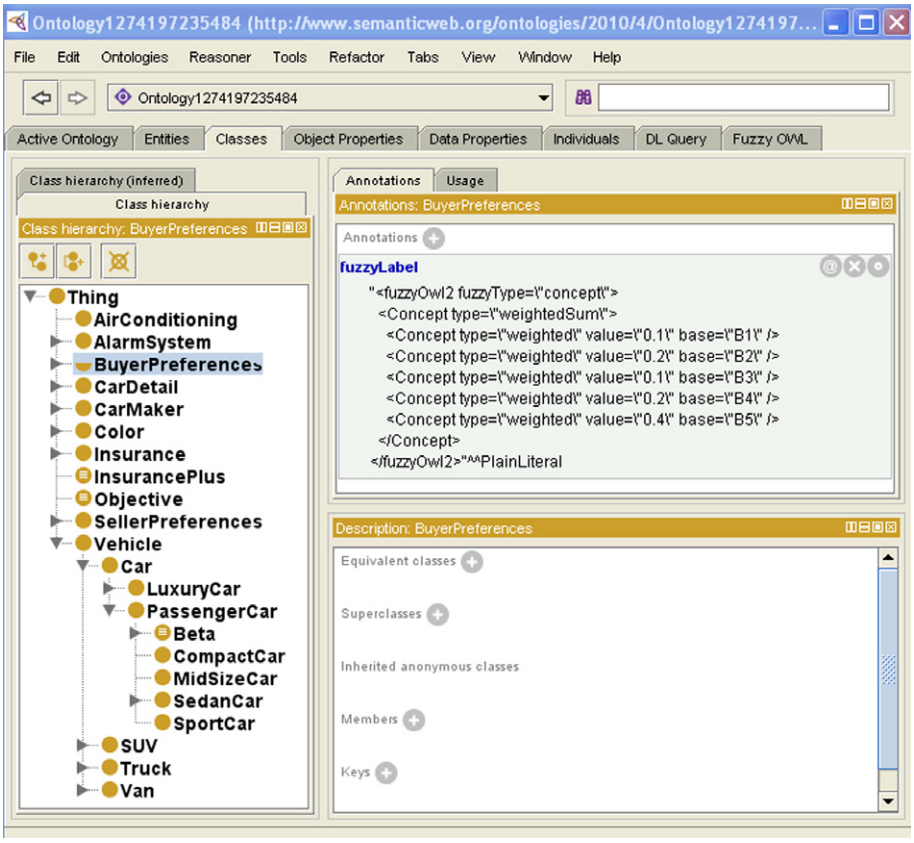


Fig. 2. Annotation property defining concept BuyerPreferences.

leq26000, ls22300–22750, ls22000–24000, and rs15000–175000 are defined datatypes with annotation properties. For instance, leq26000 represents values which are less or equal than 26000, and ls22000–24000 represents a left shoulder function with parameters $a = 22000$, $b = 24000$. This latter datatype has the following annotation property (see Fig. 3):

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

$$\text{Sell} \equiv \text{SellerRequirements} \sqcap \text{SellerPreferences}$$

$$\text{SellerRequirements} \equiv \text{SedanCar} \sqcap \exists \text{hasPrice.geq22000}$$

$$\text{S1} \equiv \neg(\exists \text{hasNavigator.NavigatorPack}) \sqcup \exists \text{hasPrice.rs225000-22750}$$

$$\text{S2} \equiv \exists \text{hasInsurance.InsurancePlus}$$

$$\text{S3} \equiv \exists \text{hasKMWarranty.SellerKmWarr}$$

$$\text{S4} \equiv \exists \text{hasMWarranty.SellerMWarr}$$

$$\text{S5} \equiv \neg(\exists \text{hasExColor.ExColorBlack}) \sqcup \exists \text{hasAirConditioning.AirConditioning}$$

SellerPreferences is represented using a weighted sum concept combining the 5 preferences of the seller (we assume that the weights of the preferences $S1$–$S5$ are 0.3, 0.1, 0.3, 0.1, 0.2, respectively). geq22000, rs225000-22750, SellerKmWarr, SellerMWarr are defined datatypes.

The ontology also includes some background information about the domain of vehicles, although this is not shown in this example.

Finally, it is clear that the best agreement among the buyer and the seller is determined by the maximal degree of satisfiability of the conjunction Buy $\sqcap$ 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 months warranty.

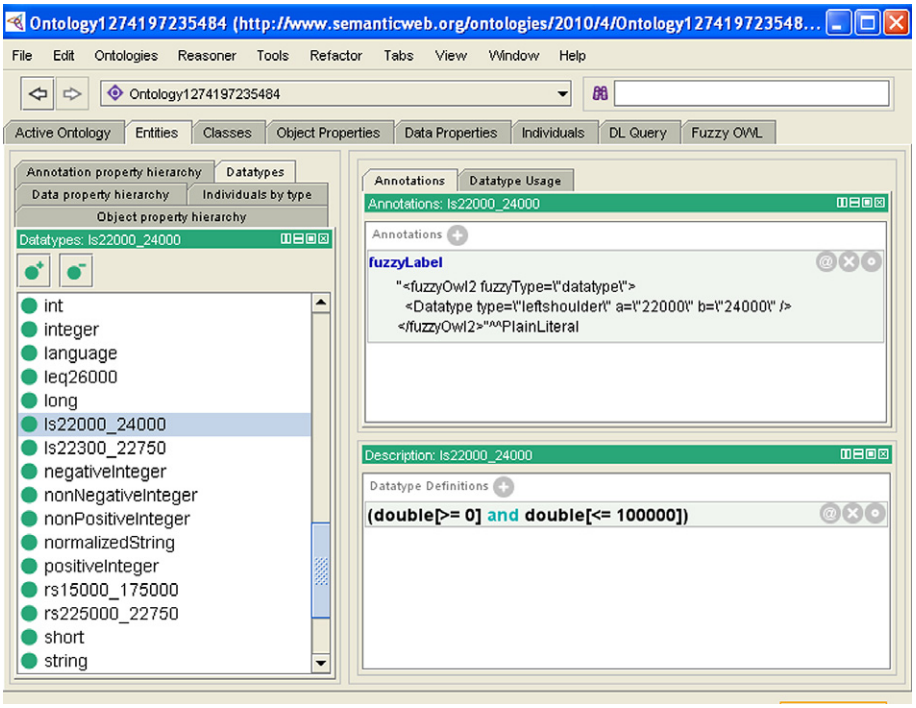


Fig. 3. Annotation property defining fuzzy datatype ls22000–24000.

4.2. Multi-criteria decision making

The following example is a modified version of the one in [55]. 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 consists in determining the optimal alternative $a^{\star}$ with the highest degree of desirability. A MCDM problem is usually expressed with a *decision matrix*, where each row corresponds to an alternative a_i, each column belongs to a criterion c_j, and the score p_{ij} describes the *performance* of alternative a_i against criterion c_j. It is possible to establish the relative importance of every criterion in the decision by assigning a *weight* to it.

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 defines its 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 a_i and for each criterion c_j by means of a *fuzzy number*, defined by means of triangular membership functions $\texttt{triangular}(a, b, c)$, which represents 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

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: Performance-211 $=$ $\exists$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.

5. Implementation

This section discusses the implementation of our approach. Section 5.1 describes a Protégé plug-in that assist users in the fuzzy ontology development process. Section 5.2 describes some parsers that translate fuzzy ontologies represented in OWL 2 into the languages supported by some fuzzy DL reasoners. Finally, Section 5.3 discusses our experimental evaluation studying the feasibility of our approach.

5.1. Editing fuzzy ontologies

Our 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] [25,42]. This allows reasoning with this part using standard ontology reasoners. Then, we can add the *fuzzy part* of the ontology by using annotation properties.

Representing the fuzzy information using OWL 2 annotations can also be done with an OWL 2 ontology editor. However, typing the annotations is a tedious and error-prone task, so we have developed a Protégé plug-in that make the syntax of the annotations transparent to the users.

The Fuzzy OWL 2 plug-in is publicly available on the web. [6] Once installed, a new tab *Fuzzy OWL* enables to use the plug-in. The plug-in has a menu with the available options (Fig. 4), which correspond to the possibilities described in Section 3. The user can choose to define fuzzy elements in the ontology (fuzzy datatypes, fuzzy modified concepts, weighted concepts, weighted sum concepts, fuzzy nominals, fuzzy modifiers, fuzzy modified roles, fuzzy axioms, and fuzzy modified datatypes), and he/she can specify the fuzzy logic used in the ontology.

Fig. 5 illustrates how the plug-in works by showing how to create a new fuzzy datatype. The user specifies the name of the datatype, and the type of the membership function. Then, the plug-in asks for the necessary parameters according to the type. A picture is displayed to help the user recall the meaning of the parameters. Then, after some basic error checkings, the new datatype is created and can be used in the ontology.

Furthermore, the plug-in is integrated with `fuzzyDL` [7] reasoner [7] and makes it possible to submit queries to it. For the moment, such queries must be expressed using the particular syntax supported by `fuzzyDL`. This allows using the reasoner without exiting Protégé, translating the annotated OWL 2 ontology into `fuzzyDL` syntax (as described in Section 5.2), and calling `fuzzyDL`.

[5] http://protege.stanford.edu/.
[6] http://www.straccia.info/software/FuzzyOWL/.
[7] http://www.straccia.info/software/fuzzyDL/fuzzyDL.html.

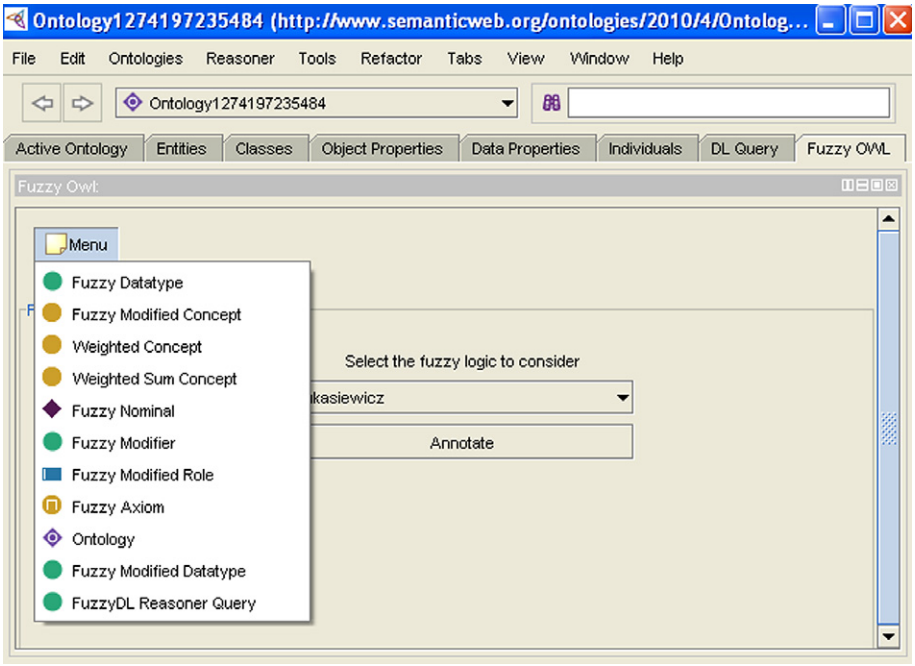


Fig. 4. Menu options of the plug-in.

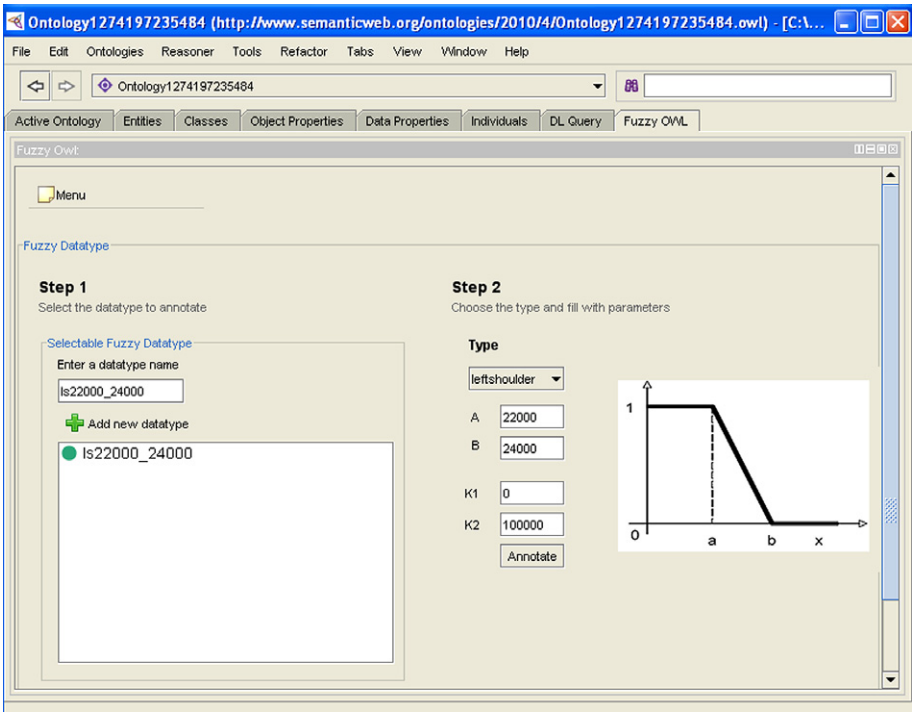


Fig. 5. Creation of a fuzzy datatype with the plug-in.

5.2. Exporting fuzzy ontologies

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.

Table 3
Fragments of fuzzy OWL 2 supported by `fuzzyDL` and `DeLorean`.

Concept	`fuzzyDL`	`DeLorean`	Axiom	`fuzzyDL`	`DeLorean`
(C1)	Yes	Yes	(A1)	Yes	Yes
(C2)	Yes	Yes	(A2)	Yes	Yes
(C3)	Yes	Yes	(A3)	No	Yes
(C4)	Yes	Yes	(A4)	Yes	Yes
(C5)	Yes	Yes	(A5)	Yes	Yes
(C6)	Yes	Yes	(A6)	No	Yes
(C7)	Yes	Yes	(A7)	No	Yes
(C8)	Yes	Yes	(A8)	Yes	Yes
(C9)	Yes	Yes	(A9)	Yes	Yes
(C10)	Yes	Yes	(A10)	Yes	Yes
(C11)	No	Yes	(A11)	Yes	Yes
(C12)	No	Yes	(A12)	Partial	Yes
(C13)	No	Yes	(A13)	Yes	Yes
(C14)	No	Yes	(A14)	Yes	Yes
(C15)	No	Yes	(A15)	Yes	Yes
(C16)	Yes	Yes	(A16)	Yes	Yes
(C17)	Yes	Yes	(A17)	Yes	Yes
(C18)	Yes	No	(A18)	Yes	Yes
(C19)	Yes	No	(A19)	Yes	Yes
Role	`fuzzyDL`	`DeLorean`	(A20)	Yes	Yes
(R1)	Yes	Yes	(A21)	Yes	Yes
(R2)	Yes	Yes	(A22)	Yes	Yes
(R3)	Yes	Yes	(A23)	No	Yes
(R4)	No	Yes	(A24)	No	Yes
(R5)	No	No	(A25)	Yes	Yes
Datatype	`fuzzyDL`	`DeLorean`	(A26)	No	Yes
(D1)	Yes	Yes	(A27)	Yes	Yes
(D2)	Yes	Yes	(A28)	No	Yes
(D3)	Yes	Yes			
(D4)	Yes	Yes			
(D5)	Yes	No			

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` [7] and `DeLorean`[8] [4]. 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 (the general parsers and the two specific parsers) are publicly available in the same web page as the Protégé plug-in. The parsers are based on *OWL API 3*[9] [23]. 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 iterating over the elements of the ontology in a transparent way. 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: it can be printed in the standard output or saved in a text file.

A full reasoning algorithm for the logic presented in Section 2.2 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]

Table 3 should not be intended as a comparison of the two reasoners. Even if `fuzzyDL` is based of fuzzy OWL Lite instead of fuzzy OWL 2, there are many features that are not available in other fuzzy DL reasoners.

5.3. Experimental evaluation

Firstly, we considered two small ontologies: the matchmaking ontology (Section 4.1) and the multi-criteria ontology (Section 4.2). As we will see, the results show that in the case of small ontologies (where not every element is fuzzy), there is no additional overhead for the annotations. It is important to stress that, due to the limited precision of measuring the running time, we have repeated the experiments 10 times and then we have computed the average result.

The matchmaking ontology has 10 annotations: 8 fuzzy datatypes (out of 14 datatypes) and 2 fuzzy concepts (out of 108 concepts). We got that the parsing time of the original matchmaking ontology is 221.8 ms, whereas the parsing time of the annotated matchmaking ontology is 219.2 ms.

The multi-criteria ontology has 13 annotations: 8 fuzzy datatypes (out of 11 datatypes) and 6 fuzzy concepts (out of 21 concepts). Now, the parsing time of the original multi-criteria ontology is 217.3 ms, whereas the parsing time of the annotated multi-criteria ontology is 217.2 ms. The parsing time of the original ontology should not be greater than the

[8] http://webdiis.unizar.es/~fbobillo/delorean.
[9] http://owlapi.sourceforge.net.
[10] `fuzzyDL` *partially* supports axioms (A12) because it restricts to the case $m = 1$.

time of the original ontology, but this results is due to the differences obtained in the different executions, and it should be interpreted as showing that there is no additional overhead for the annotations.

Then, we considered a larger ontology: *Galen*.[11] Among other elements, Galen ontology contains 23141 concepts, 25563 SubClassOf axioms, and 958 SubObjectPropertyOf axioms. We extended SubClassOf axioms and SubObjectPropertyOf axioms with a degree of truth 0.95. Furthermore, we defined some concepts as fuzzy. On the one hand, we did some experiments defining concepts as weighted concepts (WCs). On the other hand, we carried out another experiments defining concepts as weighted sum concepts (WSs) composed of 5 concepts. This latter experiment considers WSs because they require larger annotations than any other fuzzy concept.

Table 4 summarizes the results of our experimental evaluation using *Galen* ontology. Table 4c shows the influence of the percentage of annotations (%) in the parsing time (PT) and in the translation time (TT) into fuzzyDL syntax. The parsing time and the translation time are shown for both WSs and WCs concepts.

The influence of the numbers of annotated elements in the parsing time is illustrated in Table 4a. We can see that there is an approximately linear growing of the parsing time with respect to the number of elements annotated. A fuzzy ontology with a 40% of annotated elements would take 1 more second to be parsed than the original *Galen* ontology. Furthermore, we can see that in general there are no important differences between WCs and WSs, which means that the types of the fuzzy concepts are not significative.

The influence of the numbers of annotated elements in the translation time is illustrated in Table 4b. Again, there is an approximately linear growing of the running time with respect to the number of elements annotated, and there are no significant differences because of the type of the fuzzy concepts.

A translation into `DeLorean` has not been considered because that reasoner does not support WCs nor WSs, but there should not be important differences as the source codes of the parsers are very similar, with little differences due to the syntax of every language.

6. Related work

This is, to the best of our knowledge, the first effort towards fuzzy ontology representation using OWL 2, although there have been some previous attempts to represent different forms of uncertainty in ontology languages.

Some fuzzy extensions of ontology languages have been presented, more precisely OWL [19,52] and OWL 2 [51]. These languages introduce a new syntax, and thus are not within OWL 2, so current ontology editors cannot be used, as it happens under our approach. Furthermore, they are weaker from an expressive point of view since they only allow a fuzzy ABox. That is, they are restricted to our case 5 (see Section 3.1), but only for axioms (A1)–(A3). For the sake of concrete illustration, the concept assertion in Example 4 would be represented in [52] as follows:

```
<Tall rdf:about="paul" owlx:ineqType=">=" owlx:degree="0.7" />
```

Fuzzy extensions of ontology query languages have also been proposed. Notably, f-SPARQL [43] is a fuzzy extension of the query language SPARQL [44]. The authors propose the use of specially formatted SPARQL comments to specify the additional information required in the fuzzy case, namely the query type, thresholds, and the functions used in the semantics.

A closer approach to ours is [27], which uses OWL annotation properties to add probabilistic constraints, but it is restricted to our case 5, but only for axioms (A1) and (A8).

A pattern for uncertainty representation in ontologies has also been presented in [58]. However, it is restricted to our case 5, but only for axioms (A1). Furthermore, it relies in OWL Full, for which the reasoning tasks are undecidable.

Another approach is [9]. Here, annotation properties are not used, but concepts, roles and axioms are represented as individuals. For instance, the concept assertion in Example 4 would be represented using the following axioms (in abstract syntax):

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

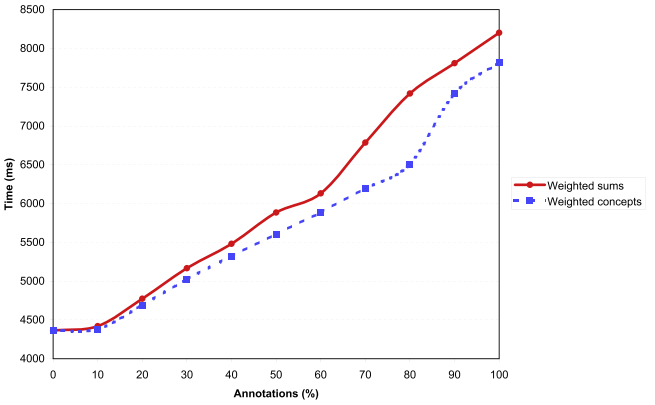
However, this representation has many problems:

- 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).
- It is not an efficient representation, since the ontology grows exponentially with the size of the ontology.

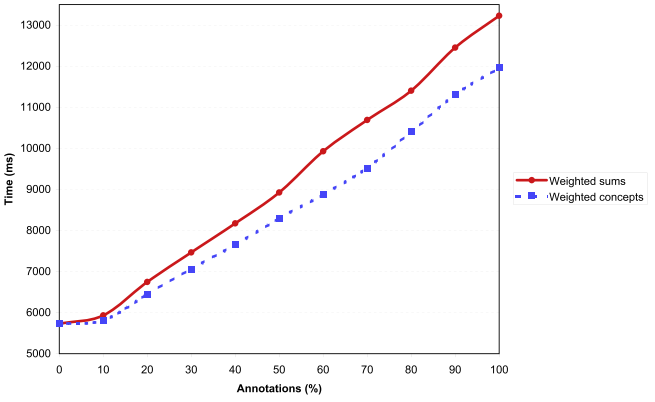
[11] www.co-ode.org/galen/full-galen.owl.

Table 4
Results of the experimental evaluation.

(a) Influence of the percentage of annotations in the parsing time



(b) Influence of the percentage of annotations in the translation time



(c) Influence of the percentage of annotations in the parsing time and in the translation time into fuzzyDL syntax

%	Concepts	GCIs	RIAs	PT WCs	PT WSs	TT WCs	TT WSs
0	0	0	0	4364.1	4363.9	5731.7	5726.1
10	2385	2604	88	4420.3	4382.5	5932.8	5812.6
20	4590	5151	177	4773.6	4692	6746.8	6443.9
30	6990	7675	276	5166.8	5025.2	7465.5	7059.4
40	9312	10152	383	5481.4	5320.3	8173.5	7648.1
50	11588	12760	462	5884.5	5603.4	8925.2	8295.4
60	13888	15260	569	6131.6	5889	9928.1	8875
70	16216	17764	672	6785.7	6193.9	10690.5	9521.6
80	18475	20363	785	7418.6	6509.4	11403.1	10402.8
90	20805	22906	875	7809.2	7418.8	12451.7	11303.3
100	23141	25563	958	8201.6	7813.8	13228.3	11962.6

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 [5,6,11,51,53]). In these works it is possible to reason using a crisp DL reasoner instead of a fuzzy DL reasoner. We instead 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. [57]. 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.

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 of ontology languages 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 considers a very general fuzzy extension of the language OWL 2, which is not simply restricted to a fuzzy ABox, but contains many other differences with respect to OWL 2, 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 reasoning with this part using standard ontology reasoners. Then, we can add the *fuzzy part* of the ontology by using annotation properties. This can also be done directly with an OWL 2 ontology editor. To this end, we have developed a graphical interface (a Protégé plug-in) making the encoding of annotation properties transparent to the user.

We have also developed some parsers translating from OWL 2 with annotations 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. Our empirical evaluation shows that our approach is feasible.

In future work, we would like to develop similar parsers for other fuzzy DL reasoners, such as `Fire`, and to improve our plug-in with new features.

Acknowledgement

We would like to thank to the anonymous reviewers for their valuable comments on an earlier version of this paper. We are also indebted to Daniele Bacarella for his help with the implementation. 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, P.F. Patel-Schneider, The Description Logic Handbook: Theory, Implementation, and Applications, Cambridge University Press, 2003.
[2] D. Beckett (Ed.), RDF/XML syntax specification, 2004. <http://www.w3.org/TR/rdf-syntax-grammar>.
[3] D. Beckett, T. Berners-Lee (Eds.), Turtle – Terse RDF triple language, 2008. <http://www.w3.org/TeamSubmission/turtle>.
[4] F. Bobillo, M. Delgado, 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), CEUR Workshop Proceedings, vol. 423, 2008.
[5] F. Bobillo, M. Delgado, J. Gómez-Romero, Crisp representations and reasoning for fuzzy ontologies, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 17 (4) (2009) 501–530.
[6] F. Bobillo, M. Delgado, J. Gómez-Romero, U. Straccia, Fuzzy description logics under Gödel semantics, International Journal of Approximate Reasoning 50 (3) (2009) 494–514.
[7] F. Bobillo, U. Straccia, fuzzyDL: an expressive fuzzy description logic reasoner, in: Proceedings of the 17th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2008), IEEE Computer Society (2008) 923–930.
[8] F. Bobillo, U. Straccia, Fuzzy description logics with general t-norms and datatypes, Fuzzy Sets and Systems 160 (23) (2009) 3382–3402.
[9] F. Bobillo, U. Straccia, An OWL ontology for fuzzy OWL 2, in: Proceedings of the 18th International Symposium on Methodologies for Intelligent Systems (ISMIS 2009), Lecture Notes in Computer Science vol. 5722, Springer-Verlag (2009) 151–160.
[10] F. Bobillo, U. Straccia, Representing fuzzy ontologies in OWL 2, in: Proceedings of the 19th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2010), IEEE Press (2010) 2695–2700.
[11] F. Bobillo, U. Straccia, Reasoning with the Łukasiewicz fuzzy description logic $\mathcal{SROIQ}$, Information Sciences 181 (2011) 758–778.
[12] S. Calegari, D. Ciucci, Granular computing applied to ontologies, International Journal of Approximate Reasoning 51 (4) (2010) 391–409.
[13] S. Calegari, F. Farina, Fuzzy ontologies and scale-free networks analysis, International Journal of Computer Science and Applications IV (II) (2007) 125–144.
[14] S. Calegari, E. Sanchez, Object-fuzzy concept network: an enrichment of ontologies in semantic information retrieval, Journal of the American Society for Information Science and Technology 59 (13) (2008) 2171–2185.
[15] P.C.G. Costa, K.B. Laskey, T. Lukasiewicz, Uncertainty representation and reasoning in the semantic web, in: J. Cardoso, M.D. Lytras (Eds.), Semantic Web Engineering in the Knowledge Society, IGI Global, 2008.
[16] B. Cuenca-Grau, I. Horrocks, B. Motik, B. Parsia, P.F. Patel-Schneider, U. Sattler, OWL 2: the next step for OWL, Journal of Web Semantics 6 (4) (2008) 309–322.
[17] S. Dasiopoulou, I. Kompatsiaris, M.G. Strintzis, Applying fuzzy DLs in the extraction of image semantics, Journal of Data Semantics XIV (2009) 105–132.
[18] S. Dasiopoulou, I. Kompatsiaris, M.G. Strintzis, Investigating fuzzy DLs-based reasoning in semantic image analysis, Multimedia Tools and Applications 46 (2) (2010) 331–370.
[19] M. Gao, C. Liu, Extending OWL by fuzzy description logic, in: Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2005), IEEE Computer Society (2005) 562–567.

[20] A. García-Cerdaña, E. Armengol, F. Esteva, Fuzzy description logics and t-norm based fuzzy logics, International Journal of Approximate Reasoning 51 (2010) 632–655.
[21] V. Haarslev, H.-I. Pai, N. Shiri, A formal framework for description logics with uncertainty, International Journal of Approximate Reasoning 50 (9) (2009) 1399–1415.
[22] P. Hájek, Metamathematics of Fuzzy Logic, Kluwer, 1998.
[23] M. Horridge, 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), CEUR Workshop Proceedings, vol. 529, 2009.
[24] M. Horridge, P.F. Patel-Schneider (Eds.), OWL 2 Web Ontology Language Manchester Syntax, 2009. <http://www.w3.org/TR/owl2-manchester-syntax>.
[25] M. Horridge, D. Tsarkov, 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), CEUR Workshop Proceedings, vol. 216, 2006.
[26] C. Hudelot, J. Atif, I. Bloch, Fuzzy spatial relation ontology for image interpretation, Fuzzy Sets and Systems 159 (15) (2008) 1929–1951.
[27] P. Klinov, 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), Lecture Notes in Computer Science vol. 5318, Springer-Verlag (2008) 213–228.
[28] R. Knappe, H. Bulskov, T. Andreasen, Perspectives on ontology-based querying, International Journal of Intelligent Systems 22 (7) (2007) 739–761.
[29] C.S. Lee, Z.-W. Jian, L.-K. Huang, A fuzzy ontology and its application to news summarization, IEEE Transactions on Systems, Man and Cybernetics, Part B 35 (5) (2005) 859–880.
[30] C.S. Lee, M.H. Wang, G. Acampora, C.Y. Hsu, H. Hagras, Diet assessment based on type-2 fuzzy ontology and fuzzy markup language, International Journal of Intelligent Systems 25 (12) (2010) 1187–1216.
[31] C.S. Lee, M.H. Wang, J.J. Chen, Ontology-based intelligent decision support agent for CMMI project monitoring and control, International Journal of Approximate Reasoning 48 (1) (2008) 62–76.
[32] C.S. Lee, M.H. Wang, H. Hagras, A type-2 fuzzy ontology and its application to personal diabetic-diet recommendation, IEEE Transactions on Fuzzy Systems 18 (2) (2010) 374–395.
[33] T. Lukasiewicz, U. Straccia, Managing uncertainty and vagueness in description logics for the semantic web, Journal of Web Semantics 6 (4) (2008) 291–308.
[34] T. Lukasiewicz, U. Straccia, Description logic programs under probabilistic uncertainty and fuzzy vagueness, International Journal of Approximate Reasoning 50 (6) (2009) 837–853.
[35] Z.M. Ma, F. Zhang, L. Yan, J. Cheng, Representing and reasoning on fuzzy UML models: a description logic approach, Expert Systems with Applications 38 (3) (2011) 2536–2549.
[36] Z.M. Ma, F. Zhang, L. Yan, Y. Lv, Formal semantics-preserving translation from fuzzy ER model to fuzzy OWL DL ontology, Web Intelligence and Agent Systems 8 (4) (2010) 397–412.
[37] T. Mailis, G. Stoilos, G. Stamou, Tractable reasoning based on the fuzzy $\mathcal{EL}++$ algorithm, in: Proceedings of the 4th International Workshop on Uncertainty Reasoning for the Semantic Web (URSW 2008), CEUR Workshop Proceedings, vol. 423, 2008.
[38] T. Mailis, G. Stoilos, G. Stamou, Expressive reasoning with Horn rules and fuzzy description logics, Knowledge and Information Systems 25 (1) (2010) 105–136.
[39] P.S. Mostert, A.L. Shields, On the structure of semigroups on a compact manifold with boundary, Annals of Mathematics 65 (1) (1957) 117–143.
[40] B. Motik, B. Parsia, P.F. Patel-Schneider (Eds.), OWL 2 Web Ontology Language XML serialization, 2009. Available from: <http://www.w3.org/TR/owl2-xml-serialization>.
[41] B. Motik, P.F. Patel-Schneider, B. Parsia (Eds.), OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax, 2009. <http://www.w3.org/TR/owl2-syntax>.
[42] N.F. Noy, M. Sintek, S. Decker, M. Crubezy, R.W. Fergerson, M.A. Musen, Creating semantic web contents with protégé-2000, IEEE Intelligent Systems 16 (2) (2001) 60–71.
[43] J.Z. Pan, G. Stamou, G. Stoilos, E. Thomas, S. Taylor, Scalable querying service over fuzzy ontologies, in: Proceedings of the 17th International World Wide Web Conference (WWW 2008), 2008, pp. 575–584.
[44] E. Prud'hommeaux, A. Seaborne (Eds.), SPARQL Query Language for RDF, 2008. <http://www.w3.org/TR/rdf-sparql-query>.
[45] T.T. Quan, S.C. Hui, T.H. Cao, Foga: a fuzzy ontology generation framework for scholarly semantic web, in: Proceedings of the Knowledge Discovery and Ontologies Workshop (KDO 2004), 2004.
[46] T.T. Quan, S.C. Hui, A.C.M. Fong, T.H. Cao, Automatic generation of ontology for scholarly semantic web, in: Proceedings of the 3rd International Semantic Web Conference (ISWC 2004), Lecture Notes in Computer Science vol. 3298, Springer-Verlag (2004) 726–740.
[47] M. Reformat, C. Ly, Ontological approach to development of computing with words based systems, International Journal of Approximate Reasoning 50 (1) (2009) 72–91.
[48] E. Sanchez (Ed.), Fuzzy Logic and the Semantic WebCapturing Intelligence, vol. 1, Elsevier Science, 2006.
[49] N. Simou, T. Athanasiadis, G. Stoilos, S. Kollias, Image indexing and retrieval using expressive fuzzy description logics, Signal, Image and Video Processing 2 (4) (2008) 321–335.
[50] G. Stoilos, N. Simou, G. Stamou, S. Kollias, Uncertainty and the semantic web, IEEE Intelligent Systems 21 (5) (2006) 84–87.
[51] G. Stoilos, G. Stamou, Extending fuzzy description logics for the semantic web, in: Proceedings of the 3rd International Workshop on OWL: Experiences and Directions (OWLED 2007), CEUR Workshop Proceedings, vol. 258, 2007.
[52] G. Stoilos, G. Stamou, J.Z. Pan, Fuzzy extensions of OWL: logical properties and reduction to fuzzy description logics, International Journal of Approximate Reasoning 51 (2010) 656–679.
[53] U. Straccia, Transforming fuzzy description logics into classical description logics, in: Proceedings of the 9th European Conference on Logics in Artificial Intelligence (JELIA 2004), Lecture Notes in Computer Science vol. 3229, Springer-Verlag (2004) 385–399.
[54] U. Straccia, Description logics with fuzzy concrete domains, in: Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI 2005), AUAI Press (2005).
[55] 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), Lecture Notes in Artificial Intelligence vol. 5711, Springer-Verlag (2009) 79–87.
[56] P. Tzouveli, N. Simou, G. Stamou, S. Kollias, Semantic classification of byzantine icons, IEEE Intelligent Systems 24 (2) (2009) 35–43.
[57] URW3, Uncertainty reasoning for the world wide web incubator group report, Technical report, W3C Incubator Group Final Report, 3, 2008. <http://www.w3.org/2005/Incubator/urw3/XGR-urw3>.
[58] M. Vacura, V. Svátek, 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), Lecture Notes in Computer Science vol. 5246, Springer-Verlag (2008) 227–234.
[59] W3C OWL Working Group, OWL 2 Web Ontology Language: Document Overview, 2009. <http://www.w3.org/TR/owl2-overview>.
[60] L.A. Zadeh, Fuzzy sets, Information and Control 8 (1965) 338–353.