Linguistic-variable Definition in OWL 2
A Protégé Plugin

Silvia Calegari¹, Davide Ciucci¹ and Matteo Mondini²
¹DISCo, Università di Milano–Bicocca, Milano, Italy
²Istituto di Ricerche Farmacologiche "Mario Negri", IRCCS, Milano, Italy

Keywords: Linguistic Variable, Fuzzy Ontology, Protégé, OWL 2.

Abstract: Linguistic Variables play a key role in the Computing with Words paradigm and in general in representing and managing vague knowledge. They are strongly related to Fuzzy Set Theory since the semantic of linguistic variables is given through fuzzy sets. In order to deal with structured vague information, it is thus fundamental to integrate ontologies and related instruments with fuzzy capabilities. Several approaches are known in literature that introduce fuzzy ontology languages, fuzzy reasoners and editor plug-ins to represent them. However, none directly deals with the problem of representing linguistic variables. In the present paper, we introduce a Protégé Plugin developed to define linguistic variables in ontologies. The plug-in is based on OWL2 and on a lite version of its fuzzy extension.

1 INTRODUCTION

In the last decade the need to represent and share knowledge in several domain applications has given rise to a growing interest in research on ontology. An ontology is defined as an explicit specification of a conceptualization (Gruber, 1993a; Gruber, 1993b) and following this principle an ontology identifies the objects (both abstract and concrete things of the domain) and the relations that link them. Ontologies are used in many research areas such as knowledge engineering, database design, information retrieval and extraction, context-aware, agent-based system, etc. (Guarino, 1998); but its great notoriety can be awarded to the key role in the Semantic Web.

In the Semantic Web a critical open issue is how to deal with imprecise and vague knowledge that is a typical factor in real world applications (Sanchez, 2006). It is well known that Fuzzy Set Theory is able to describe vague concepts through a generalized notion of set according to which an object may belong to it with a certain degree (typically, a real number in the range [0, 1]) (Klir and Yuan, 1996). Many researchers have focused their research activities in order to integrate the fuzzy logic formalism in the ontological one with the aim of defining the so called Fuzzy Ontology (Sanchez and Yamanoi, 2006; Calegari and Ciucci, 2008; Calegari and Sanchez, 2008). To support the fuzzy ontology formalism is a really hard topic that mainly includes four aspects: (i) extend both the semantic and syntax of Description Logics¹ (DLs) with fuzziness, (ii) define a fuzzy extension of the OWL language (OWL, 2005) (the well-known ontology language that provides three increasingly expressive languages, i.e. “OWL Lite”, “OWL DL” and “OWL Full”) based on the considered fuzzy DL, (iii) have reasoners that can support the fuzzy OWL language, and (iv) extend ontology editors with new plug-ins able to define and represent fuzzy knowledge based on the previous points.

In literature, there exist many papers related to the definition of fuzzy DLs (Bobillo and Straccia, 2013; Bobillo et al., 2013b; Mailis et al., 2010) or fuzzy OWL (Calegari and Ciucci, 2007b; Gao and Liu, 2005; Stoilos et al., 2005), but none of these works do tackle the problem of real situations where experts by interacting with an ontology editor have to concretely define knowledge based on the fuzzy ontology formalism. Also to define fuzzy reasoners is not a simple process due to the time complexity for the execution of some reasoning tasks. Currently only the fuzzyDL² reasoner for fuzzy $\mathcal{SH I F}$ with concrete fuzzy concepts has been presented. This rea-
soner supports the “OWL Lite” language which is the less expressive OWL language as it is used only for a classification hierarchy and simple constraints features. At this point, straightforward problems on the use of scalable reasoning procedures to deal with vague knowledge can arise too (Cimiano et al., 2008). To overcome the scalability issue, a trend of research deals with how to reduce fuzzy DLs to crisp DLs with the aim of preserving the knowledge satisﬁability. In this way, standard DL reasoners based on the crisp OWL language can be used for reasoning with fuzzy ontologies. For example, in (Bobillo et al., 2012; Bobillo et al., 2013b) the Pellet\(^1\)(Sirin et al., 2007)) DL reasoner that supports OWL 2\(^2\) is adopted. OWL 2 is the new standard ontology language for the Semantic Web supported by the W3C\(^3\), and it is conformed to the “OWL 2 DL” language.

The aim of this paper is to present the development of a plug-in, called FuzzyOntologyView, that is compliant with the fuzzy logic formalism. The Protégé 4.2\(^4\) ontology editor has been chosen to be integrated with the FuzzyOntologyView plug-in as it supports OWL API 3\(^5\) (Horridge and Bechhofer, 2011), the latest version of the OWL APIs. Our plug-in does not use fuzzy (or crisp) reasoners but it allows to manage fuzzy knowledge by deﬁning fuzzy concepts that are represented as linguistic variables in the sense of Zadeh (Zadeh, 1975). A linguistic variable is a variable whose values correspond to linguistic adjectives, e.g. the variable Height with values “small, medium and tall”. We propose a solution to extend the OWL 2 language by including fuzzy concepts deﬁned as linguistic variables preserving the standard formalism of OWL 2\(^6\). As previously stated, the works presented in (Bobillo et al., 2012; Bobillo et al., 2013b) propose a solution to reduce the fuzzy ontology to the corresponding crisp ontology using standard DL reasoners. To this aim, a plug-in in Protégé 4.2 (Bobillo and Straccia, 2011) that allows to manage fuzzy knowledge has been developed, but only by adding such information as annotations simplifying the problem of representing the fuzzy knowledge in OWL 2.

On the other hand, the plug-ins deﬁned in (H. Ghorbel and Ciucci, 2007a) are able to manage fuzzy lite ontologies where only membership relations are considered, and they support old ontology languages, i.e. RDFS and “OWL Lite”, respectively. The paper is organized as follows: Section 2 introduces the fuzzy notions used for the development of the plug-in, in Section 3 the FuzzyOntologyView plugin is presented and comparative evaluations with other solutions proposed in literature are given. In Section 4 some conclusions and possible future trends are outlined.

2 FUZZY ONTOLOGY

2.1 Fuzzy Set Basics

Fuzzy sets on a universe \(U\) are a generalization of standard sets defined through a membership function \(f_A : U \rightarrow [0,1]\) which associates with any element \(x \in U\) its degree of membership to a set \(A\). Fuzzy set theory has almost ﬁfty years, the ﬁrst paper dates back to 1965 (Zadeh, 1965), and it is now widely used in all those situations where we need to represent and manage vague information. Despite of this general deﬁnition of a fuzzy set, where no restrictions are put on the function \(f_A\), the membership function shape is often chosen among a limited set of possibilities. The most used ones are: singleton, triangular, trapezoidal, gaussian and beta, whose deﬁnition reads as follows.

- **Singleton**: \(s : U \rightarrow [0,1]\), there exists a particular element \(x \in U\) such that \(s(x) = 1\) and \(s(y) = 0\) for all \(y \neq x\), that is the set \(s\) has only one element with membership value equal to 1;

- **Trapezoidal**: \(Tr : U \rightarrow [0,1]\), \(Tr(x) = \max(\min(\frac{x-a}{b-a}, \frac{d-x}{c-d}), 0)\) where \(a,b,c,d\) are constants whose meaning is clear from ﬁgure 1;

- **Triangular fuzzy sets** are trapezoidal ones with \(b = c\);

- **Gaussian**: \(G : U \rightarrow [0,1]\), \(G(x) = e^{-\frac{(x-c)^2}{2\sigma^2}}\) with \(c\) the center of the curve and \(\sigma\) the width;

- **Beta**: \(B : U \rightarrow [0,1]\), \(B(x) = x^{(\alpha-1)}(1-x)^{(\beta-1)}\) for \(\alpha, \beta > 0\) real-valued parameters.

\[\text{Figure 1: An example of trapezoidal fuzzy set.}\]

---

\(^1\)http://clarkparsia.com/pellet
\(^2\)OWL 2 Web Ontology Language: http://www.w3.org/TR/owl2-overview/
\(^3\)World Wide Web Consortium: http://www.w3.org/
\(^4\)http://protege.stanford.edu/download/registered.html#p4.2
\(^5\)http://owlapi.sourceforge.net
\(^6\)Let us remark that we need some functionalities of OWL2 not present in OWL, such as punning and extended datatypes
We now introduce linguistic variables (Zadeh, 1975) a central notion in so called Computing with words paradigm (Zadeh, 1996). A linguistic variable is a quintuple \((E, T(L), U, G, M)\) where \(L\) is the name of the variable, \(T(L)\) is the set of values for \(L\), which are called linguistic terms (also linguistic values or linguistic labels), \(U\) is a universe of discourse; \(G\) is a syntactic rule which generates the terms in \(T(L)\), and \(M\) is a semantic rule which maps each linguistic term \(A \in T(L)\) to its meaning, \(M(A)\), where \(M(A)\) denotes a fuzzy subset of \(U\).

**Example 2.1.** As an example let us consider to define the concept of Height with its nuances. We have that the linguistic variable is \(L = \text{Height}\) and its values are \(T(L) = \{\text{small}, \text{medium}, \text{tall}\}\). The universe is the interval \(U = [0, 200]\) expressing the height in centimeters, the rule \(G\) just says that the only linguistic terms are the ones in \(T(L)\) and the mapping \(M\) is represented by the fuzzy subsets of Figure 2.

![Figure 2: “Height”: an example of Linguistic Variable.](image)

### 2.2 Combining Fuzzy Sets with Ontologies

As said in the introduction, several approaches to use fuzzy sets with ontologies can be found in literature. Here, we are dealing with a simple situation, where each concept is a fuzzy subset on the instance domain.

**Definition 2.1.** A Fuzzy Ontology is defined as the tuple \((I, C, R, F, A)\) where:

- \(I\) is the set of individuals, also called instances.
- \(C\) is the set of concepts. Each concept \(C \in C\) is a fuzzy set on the domain of instances \(I : I \rightarrow [0, 1]\).

A special role is held by the taxonomic relation \(T \subseteq C^2\) which identifies the subsumption relation among the concepts.

- \(R\) is the set of non-taxonomic and crisp relations;
- \(F\) is the set of the fuzzy relations on the set of entities \(E\) and a specific domain contained in \(D = \{\text{integer, string, ...}\}\). In detail, they are \(n\)-ary functions such that each element \(F \in F\) is a relation \(F : \mathbb{I}^{(n-1)} \times P \rightarrow [0, 1]\) where \(P \in D\).
- \(A\) is the set of axioms expressed in a proper logical language, i.e. predicates that constrain the meaning of concepts, individuals, relationships and functions.

This corresponds to a fragment of the more general definition of fuzzy ontology (Calegari and Ciucci, 2008), where any relation can be fuzzy and more freedom is given in the possibility to define axioms. This issue becomes more clear when considering the OWL/DL definition of fuzzy ontology. Indeed, from a more formal standpoint we have to consider the definition of a fuzzy ontology based on description logic. Let us note that in literature several definitions of Fuzzy Description Logic can be found (Straccia, 2006; Stoilos et al., 2005; Calegari and Ciucci, 2007b; Bobillo et al., 2012). We consider here a fuzzy variant of \(SROIQ(D)\), since it is equivalent to the OWL2, the W3C standard language. However, in order to cope with linguistic variables only a fragment of fuzzy \(SROIQ(D)\) is needed. The only parts which need to be fuzzified are concepts, concrete domains and individual axioms. Indeed, in order to represent the concept \(\text{Small-People}\), subconcept of \(\text{People}\) (i.e., the set of people corresponding to the linguistic term \(\text{Small}\), relative to the linguistic variable \(\text{Height}\)), we need to define the relation \(\text{Height}\), concrete fuzzy concept \(\text{Small}\) as a fuzzy set on the universe \([0, 250]\) and then state that the concept \(\text{Small-People}\) is equivalent to \(\text{Height}\). Small. For further details about \(SROIQ(D)\) we refer the readers to (Bobillo et al., 2012).

### 3 FuzzyOntologyView: THE PROTÉGÉ PLUGIN

This section presents the Protégé plugin that integrates the Fuzzy Set Theory notions (see Section 2) into the ontology-knowledge; this goal is obtained by defining linguistic variables through fuzzy concepts. The first part of this section describes the main phases of the FuzzyOntologyView plug-in that allow to create the fuzzy concepts. The second part explains how the fuzzy concepts definition can preserve the standard formalism of OWL 2 and provides a simple example. The third part gives a comparison of the FuzzyOntologyView plug-in with other approaches proposed in the literature.

#### 3.1 Defining Fuzzy Knowledge in Domain Ontologies

To interact with the plug-in, it must be previously installed in the usual way of Protégé plug-ins. A new tab called FuzzyOWLView will be added as a new functionality of the standard element Classes in the editor, i.e., beside the tabs Class hierarchy and Class...
**hierarchy (inferred).** The first two phases for using the *FuzzyOWLView* tab are: (1) create/import the non-fuzzy domain Ontology, (2) select the concept from the *Class hierarchy* tab that has to be fuzzified. Now, a user can select the *FuzzyOWLView* tab and choose the membership function (see Section 2) that better represents the semantic of the fuzzified concept.

As an example we define inside the *Travel* domain Ontology the linguistic variable *Airport Distance*, the two terms *Near Airport*, *Not so far Airport* and finally, the individuals *Roma*, *Madrid*, *Washington* will be associated to these terms.

However, before it is mandatory to choose the *DataProperty* associated with the fuzzy concept, otherwise an alert message will advice the user. In the example, the *DataProperty* is *AirportDistance* and the fuzzy concept is *nearAirport*.

![Figure 3: Selection of a membership function.](image)

Figure 3 shows the *FuzzyOWLView* tab where the concept *nearAirport* has been selected and the list of four membership functions that a user has to choose among. In our example, we suppose that the *Trapezoid* function has been selected, and the related panel is displayed. Figure 4 illustrates the *Trapezoidal* function defined by typing the values of the constants in the specific area called *Trapezoid parameters*.

After that a membership function is associated with a concept, whenever the same concept is chosen by interacting with *FuzzyOWLView* then a new panel will be displayed to the user (see Figure 5). Now, there are three different operations on individuals of a fuzzy concept: (1) create a new individual to associate with the concept, (2) add an existing individual in the ontology to the concept by selecting the individual from a list and then write a suitable value, and (3) view all the fuzzy concepts and individuals that belong to a concept.

![Figure 4: Trapezoidal function for the concept nearAirport.](image)

![Figure 5: Management of individuals to a fuzzy concept.](image)
have been assigned to the same DataProperty.

In Figure 6 a user is creating the individual Roma belonging to the fuzzy concept not-so-farAirport with a distance from the airport of 28km. As it happens for the creation of a membership function, a user can draw and re-draw an individual and only the selection of the button Create individual establishes its definition in the Ontology.

In Figure 7 there is a more complete view of the fuzzy concepts and individuals that have been defined. In this case, for the DataProperty AirportDistance two fuzzy concepts have been defined, i.e. nearAirport and not-so-farAirport, and two individuals have been assigned to them, i.e. Madrid and Washington cities with the meaning that Madrid is more near to its airport than Washington (considering Dulles International Airport).

### 3.2 Translation in OWL 2

The output of the FuzzyOntologyView plug-in is a fuzzy ontology that can be directly encoded in the OWL 2 language by the usage of the library OWL API 3. The crucial objective of adding fuzzy notions by preserving the standard formalism of OWL 2 will be reached by a suitable definition of linguistic variables. In the following all the elements defined to map the notion of linguistic variable in OWL 2 are explained. These elements act on Classes, ObjectProperties and DataProperties. Classes:

- **FuzzyTypeDefinition.** This class can be considered one of the most important as its scope is to define ternary relations among an individual, a concept and its degree of membership. In OWL 2 it is not possible to directly create a ternary relation among a concept, an individual and a degree. The solution adopted in the ontology languages is to consider the standard notion of reification where a link between two individuals or an individual and a value is defined. This means to duplicate the concept involved in the relation as an individual and to link this last with each instance assigned to the starting fuzzy concept (see Section 3.1). By considering the example of Figure 7 and the notion of linguistic variable (see Section 3), we have four ternary relations between two individuals and their degree as each individual belongs to the two fuzzy concepts (NearAirport, not-so-far-Airport) with a different degree. For example, one of them is \( r_1(\text{Madrid}, \text{nearAirport} + \text{code}) = 1.0 \) with the assumption that the transformation ‘concept \( \mapsto \) individual’ for nearAirport is applied. During the reification process a code is automatically added to the new instance by the editor. Finally, the class FuzzyTypeDefinition is defined at the top level in the hierarchy, and when this class is selected in the Usage tab all the ternary relations between individuals are listed.

- **Type of membership function.** Several classes related to the type of membership function selected by the user are instanciated. In the example of Section 3.1 the class TrapezoidalMembershipFunction is defined at the top level in the hierarchy tree as the user has always preferred the Trapezoidal function, and thus only the class related to this function has to be presented to the user. In case of selection of other functions, the corresponding class will be instanciated as GaussianMembershipFunction, SingletonMembershipFunction, and BetaMembershipFunction. In the considered example, when the TrapezoidalMembershipFunction class is selected all the Trapezoidal functions are listed in the Usage tab. Moreover, at this step, the definitions of the specific Trapezoidal functions created in the ontology are shown. By considering the example of Section 3.1, the functions membershipFunctionOfnearAirport and membershipFunctionOfnot-so-farAirport are listed as individuals. This is due to the necessity to manage and save the parameters inserted by the user during the creation of the specific membership function (see the statement datatype in DataProperties).
FuzzyConcept. This class is defined to know all the concepts that have been defined as a membership function in the ontology, and it is created at the top level in the hierarchy. By considering the example of Section 3.1, the two concepts nearAirport and not-so-farAirport defined as individuals (see FuzzyTypeDefinition) are listed.

ObjectProperties:
- fuzzyType. It represents the name of the relation between an individual and the FuzzyTypeDefinition class.
- hasBaseType. It links a FuzzyTypeDefinition to a concept in order to represent which individuals belong to the specific concept.
- hasMembershipFunction. It links a concept to the related fuzzy membership function.

DataProperties:
- hasDegree. It indicates the degree between two individuals in a ternary relation as defined in the FuzzyTypeDefinition class. The degree is defined in the range [0, 1]. This value is automatically obtained as shown in the example of Figure 2 where the compatibility of Height 156cm with Small might be 0.35.
- datatype. It allows to save the parameters for the type of membership function selected by the user.
- DataProperty. It indicates the DataProperty name used to group all the fuzzy concepts that have a semantic association (i.e., the linguistic variable name). Moreover, by selecting each individual it also stores the value that has been written by the user during the creation of the individual for a specific fuzzy concept (see Section 3.1 where the user has inserted the value 12Km to indicate the distance from the city of Madrid to its airport).

3.2.1 From a Linguistic Variable to OWL 2

We now explain through an example how a linguistic term has been represented in OWL 2.

First of all, a linguistic variable L, AirportDistance in our case, is realized through a DataProperty. Then, a term A ∈ T(L) is a concept, in our example Not-so-farAirport is a subclass of the concept City. In order to represent the semantic of a term, that is the fuzzy set M(A), we reify the term A = Not-so-farAirport (see FuzzyTypeDefinition) and obtain the individual Not-so-farAirport which belongs to the class FuzzyConcept and is linked through the object property hasMembershipFunction to the individual MembershipFunctionOfnot-so-farAirport. This last individual, on its turn, is a member of the class TrapezoidalMembershipFunction. The four values a, b, c, d defining the trapezoidal shape of the fuzzy set M(Not so farAirport) are set using four datatype properties HasFirstParameter, HasSecondParameter, HasThirdParameter, HasFourthParameter of MembershipFunctionOfnot-so-farAirport.

Finally, we explain how to represent individuals. Let us consider the case of Madrid, whose airport is 12Km from the city and thus its membership degree to not-so-farAirport is 0.0. The distance in kilometers, 12, is the value of the property AirportDistance whereas 0.0 is the value of the property hasDegree of the individual Fuzzynot-so-far-Airport-921045970, linked to Madrid through the fuzzyType property. Let us note that the individual Fuzzynot-so-far-Airport-921045970 is automatically created by the editor and generally is the value displayed during the reification process where the code 921045970 is added as previously explained. In order to make the semantic of each individual readable, we have defined a reified ternary relation between the name of the individual and the individual having assigned the automatically generated code by the object property fuzzyType. By considering the same example, we have fuzzyType(madrid,Fuzzynot-so-farAirport-921045970).

Here below, a partial fuzzified knowledge in OWL 2 of the Travel Ontology related to the example explained in the paper is reported. In particular, we consider the definition of the city Madrid and its distance to the airport by analysing the belongingness of the fuzzy concept not-so-farAirport.

```xml
<!-- Individual: Individual: Madison
<Declaration>
  <NamedIndividual IRI="#madrid"/>
</Declaration>
</ClassAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="#AirportDistance"/>
  <NamedIndividual IRI="#madrid"/>
  <Literal datatype=IRI="#xsd:float">12.0</Literal>
</DataPropertyAssertion>
</Individual>
</Individual>
</Individual>
</Individual>
</Individual>
```
Figure 8 shows some of the fuzzy notions previously described by clicking-on the individual Madrid after the selection of the concept not-so-farAirport. As expected and depicted in Figure 7, two fuzzy-Type are defined, i.e. fuzzyType 1.0 nearAirport and fuzzyType 0.0 not-so-farAirport. In the lower right window of Figure 8 also the DataProperty information is reported where the name of the property (AirportDistance) and the value wrote by the user (12.0f) are stated (the value is stored as a float number). The same information is also reported in the Usage tab for Individuals. At the bottom left part of the figure, the concepts FuzzyConcept, FuzzyTypeDefinition are located at the top level of the hierarchy with the meaning explained previously. Let us note that when the Individuals tab is focused on the menu, then the FuzzyOWLView tab is displayed as a plug-in and not as a tab beside the ones defined for the Classes tab as shown in Figure 5.

3.3 Other Solutions

The solution we adopted to represent a linguistic variable, and specifically a fuzzy concept, is not the only possible one in OWL2. Other possibilities are:

- Add a Datatype property to the fuzzy axioms in order to represent the membership degree. However, in this case it would be impossible for an object to belong to two different classes, since we would have two “withDegree” tags for the same object without knowing the linked property, as in the following example:

```xml
<fowl:Thing rdf:about="a">
  <rdf:type rdf:resource="A"/>
  <rdf:type rdf:resource="B"/>
  <fowl:withDegree>0.5</fowl:withDegree>
  <fowl:withDegree>0.4</fowl:withDegree>
</fowl:Thing>
```

- Subclasses. Create a subclass of a fuzzy concept C for any desired degree C0.1, C0.2, etc. This solu-
tion requires either to parse the name of the concept to know the membership degree or to add it as a property to the concept $C$, a feature only present in OWL full (which is not desirable to use for its well-known undecidability).

- Annotations. With this solutions all the fuzzy part is realized through the annotations. In this way all the fuzzy constructs and axioms can be represented and standard editors and reasoners can use the classical part. However, this is not compliant with the scope of the annotations which, according to the W3C, “should be used when the information attached to entities should not be considered a part of the domain and when it should not contribute to the logical consequences of an ontology”. Indeed, it is desirable that the fuzzy elements are at the same semantic level of the standard ones.

This last solution is the only one implemented in literature by (Bobillo and Straccia, 2011). It is part of a complete suite with a Fuzzy Reasoner which enables to represent and reason with a Fuzzy Ontology. Besides the fact the we are moving towards a real fuzzy ontology (not a one existing only in comments) as described above, the main improvements to represent fuzzy concepts introduced in our solution can be summarized as follows:

**Representation and Building of Linguistic Variables:**

- The possibility to represent a linguistic variable, with all the linguistic terms linked among them (see Section 3.1);
- The simplicity to represent a linguistic term. Indeed, in the approach (Bobillo and Straccia, 2011) in order to create the concept “City not so far from airport” we have to
  - Create the DataProperty $\text{Has-Distance-to-airport}$ and set it as a superclass of the concept City;
  - Create the trapezoidal fuzzy datatype $\text{not-so-far-airport}$;
  - Add the concept $\text{City-no-far-airport}$ with an annotation that it is a fuzzy concept, using a specify fuzzy logic (Zadeh or Łukasiewicz);
  - State that $\text{City-no-far-airport}$ is equivalent to a city with $\text{Has-Distance-to-airport SOME not-so-far-airport}$;

A list of operations which is definitely more complex than in our plug-in.

**Graphical Issues:**

- the dynamical graphic representation, which is absent in the Bobillo-Straccia approach. Indeed, in our plug-in it is possible to draw and re-draw infinite time a function before to create it in the ontology; instead in the Bobillo-Straccia approach only a static picture is displayed to help the user recall the meaning of the parameters but a user does not have the view of the membership function created. In addition, at any time it is possible to have a complete overview of all the fuzzy concepts and individuals associated with a specific DataProperty. This functionality can improve the usability of the system by supporting the user during the definition of the domain Ontology.

  - in the Bobillo-Straccia approach only a few membership functions can be defined such as triangular, trapezoidal and linear.
  - the ontology updates in real-time when some modifications occur, whereas in the Bobillo-Straccia approach it is necessary to save the ontology and re-open it.

**4 CONCLUSIONS AND FUTURE WORK**

This paper has presented the FuzzyOntologyView plug-in, the first that integrates the notion of linguistic variable in ontologies. FuzzyOntologyView has been defined for the well known ontology editor Protégé. In detail, the plug-in is compliant with the last version of this editor that directly supports the OWL 2 language. To extend the ontology knowledge with fuzzy notions that is directly supported in OWL 2 is not an easy task. Indeed, in the literature a solution (Bobillo et al., 2012; Bobillo et al., 2013b) that allows to manage fuzzy knowledge has been presented, but only by adding such information through the statement annotation in OWL 2, thus simplifying the problem of a direct representation in OWL 2.

In future works, we plan to integrate the DE-LOREAN reasoner inside the FuzzyOntologyView plug-in with the objective to directly use all the definable fuzzy notions to infer new knowledge inside the OWL2 framework. Once this step will be complete, the plug-in will be made available on-line.

**REFERENCES**


