Fuzzy Protégé for Fuzzy Ontology Models

Hanêne GHORBEL --- Afef BAHRI --- Rafik BOUAZIZ
MIRACL Laboratory, Faculty of economics and management of Sfax - Tunisia
E-mail : Hanen.Ghorbel@fsegs.rnu.tn ; afef.bahri@gmail.com ; raf.bouaziz@fsegs.rnu.tn

Abstract. *Fuzzy Protégé* is a semi-automatic collaborative tool for the construction of fuzzy ontology models, built as a Protégé 3.3.1 tab plug-in. *Fuzzy Protégé* is an extension of the well known ontology editor *Protégé* for which we have defined new meta-classes to allow the definition of parameterized membership functions. *Fuzzy Protégé* also gives support to instantiate fuzzy concepts and roles and allows automatic computing of membership degrees. Finally, *Fuzzy Protégé* allows querying fuzzy ontologies based on fuzzy criteria. We present in this paper the internal architecture of *Fuzzy protégé* and we give some details on its implementation and the way we use it to build and validate fuzzy ontologies.

Key words. Semantic Web, Fuzzy Ontologies, *Fuzzy Protégé*, Inference in Fuzzy Ontology, Interrogation for Fuzzy Ontologies.

1 Introduction

The web content is, for the most part, subject to imperfection (i.e. imprecision, uncertainty, ambiguity). Semantic Web ontologies are based on crisp logic and do not provide well-defined means for expressing imperfection. How to represent non crisp data, such as “an expensive book”, “a good customer” within the ontology definition. The idea of fuzzy set and fuzzy logic theory was first proposed by Zadeh [Zadeh, 75], as a mean of handling uncertainty. The appearance of a wide variety of methodologies for the construction of fuzzy membership functions motivates the application of fuzzy set theory in a great number of application fields. In this paper, we talk about the integration of fuzzy logic in ontology in order to define which we call fuzzy ontology. Ontology tools are most based on crisp logic and do not provide well-defined means for expression fuzzyness. We propose in this paper a framework for fuzzy ontology building, *Fuzzy Protégé*, as an extension of the well-known ontology editor *Protégé*. Then we describe how *Fuzzy Protégé* has been designed an implemented. The paper is structured as follows. Section 2 describes the internal architecture of the framework and the implementation details of the building modules, showing how to use *Fuzzy Protégé* to define fuzzy ontology components. Section 3 defines a rule inference engine implemented with Jess. Section 4 presents a flexible query tool. Finally, in section 5, we conclude and present some topics for further research.

2 Architecture of Fuzzy Protégé

Fuzzy ontology has been introduced by Straccia [06] to represent knowledge in all domains in which the concepts to be represented have imprecise definitions. Fuzzy concepts and roles are considered as fuzzy sets. Thus an instance does not fully belong or not to a given fuzzy concept (resp. fuzzy role) but has a membership degree being an instance of that concept (resp. role). For instance, the fuzzy concept “Young_Person” is defined as follow:

$$\text{Young\_Person} = \text{Person} \cap \exists\ \text{Age.young}$$

The linguistic term Young may be defined by a trapezoidal function as shown graphically in Figure 1. Mathematically, it is defined as follows:

$$\mu_{Young}(\mathbf{Age}) = \begin{cases} (Age - 12) / (18 - 12) & \textit{if } 12 <= Age <= 18 \\ 1 & \textit{if } 18 < Age <= 30 \\ (45 - Age) / (45 - 30) & \textit{if } 30 < Age <= 45 \\ 0 & \textit{if } Age < 12 \textit{ Or } Age > 45 \end{cases}$$

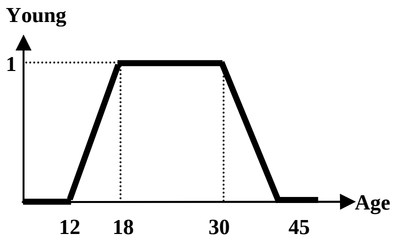

Fig.1. Example for trapezoidal membersh ip function

The principle idea of our framework is the use of parameterized membership functions to allow automatic attribution of membership degree. Then we will be able to perform inconsistency checking as membership degrees are not randomly added as some works do. We define a new metaclass named “Fuzzy_Class” in the *Protégé* metaclasses hierarchy (basic class is the class “Thing”). The figure 2 shows the metaclasses hierarchy of *Fuzzy Protégé*. To use these metaclasses, we have to import the specific meta-data using Meta data plug-in. So, the “Fuzzy Meta-Classes” become classes systems, users of the framework can neither modify nor remove them (cf. Figure 3). Indeed, users may define a given fuzzy ontological component as a subclass of one of the Fuzzy classes defined in the model (i.e. Fuzzy_Class_Ti, Fuzzy_Class_Trz, Fuzzy_Class_L, Fuzzy_Class_R).

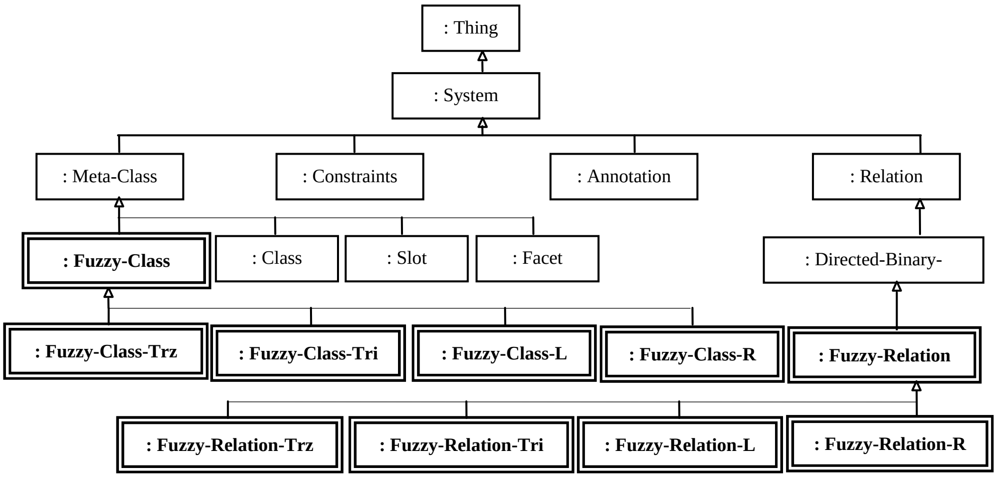

Fig.2. Fuzzy *Protégé* Hierarchy metaclasses

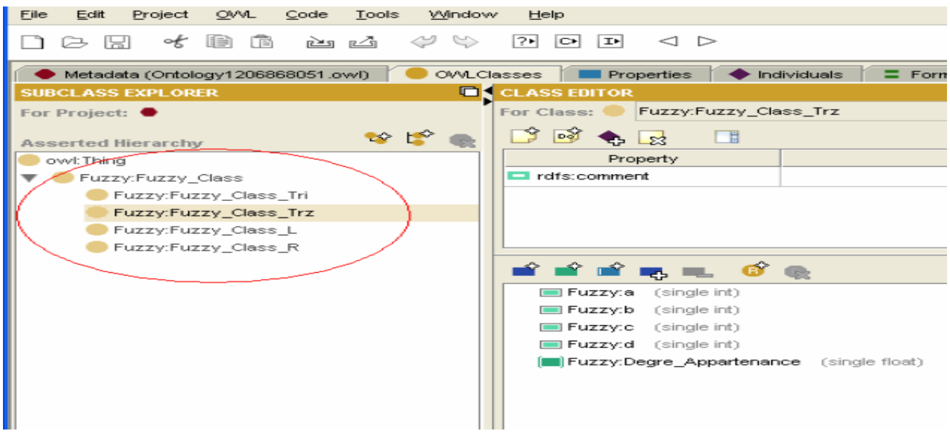

Fig.3. Meta-classes of ***Fuzzy Protégé***

3 Inference and validation of fuzzy instances

Instantiation is considered as a type of inference. In the case of fuzzy ontologies, instantiation consists on determining the degrees of membership of instances to fuzzy concepts or roles. We define in *Fuzzy Protégé* a rule based inference engine for automatic membership degree computing. The module is implemented under the *Jess* language and we use the *Jess Tab* for compilation. We give in the following, the definition of rules used for membership degree computing of a concept defined with trapezoidal membership function.

Rule 1: Determinate of degree of membership if a <= X <= b

```
(if (and(slot-get <--Instance--> X >= Fuzzy_a) (slot-get <--instance--> X <= Fuzzy_b)) then (slot-set <--Instance-->
Fuzzy:Degree_Of_Membership (/ (- (slot-get <--Instance--> X) Fuzzy_b)) (- (Fuzzy_b – Fuzzy_a))) (if (and(slot-get <--
Instance--> X >= Fuzzy_b) (slot-get <--instance--> X <= Fuzzy_c)) then (slot-set <--Instance-->
Fuzzy:Degree_Of_Membership (- (slot-get <--Instance--> X) Fuzzy_b)))
```

Rule 2: Determinate of degree of membership if b < X <= c

```
(if (and(slot-get <--Instance--> X > Fuzzy_b) (slot-get <--instance--> X<= Fuzzy_c)) then (slot-set <--Instance-->
Fuzzy:Degree_Of_Membership 1.0)
```

Rule 3: Determinate of degree of membership if c < X <= d

```
(if (and(slot-get <--Instance--> X > Fuzzy_c) (slot-get <--instance--> X <= Fuzzy_d)) then (slot-set <--Instance-->
Fuzzy:Degree_Of_Membership (/ (+ ( - slot-get <--Instance--> X) Fuzzy_d)) (- (Fuzzy_d – Fuzzy_c)))
```

Rule 4: Determinate of degree of membership if X > d And X < a

```
(if (and(slot-get <--Instance--> X <= Fuzzy_a) (slot-get <--instance--> X >= Fuzzy_d)) then (slot-set <--Instance-->
Fuzzy:Degree_Of_Membership 0.0))
```

4 Fuzzy ontology querying

Fuzzy Protégé contains equally a module for fuzzy ontology querying. When a user asks a query about the instances of a given fuzzy concept, we have to return those instances that have a membership degree to the corresponding concept great or equal to a given threshold. In order to do that, we propose to realize a pretreatment on the instances of fuzzy concepts before query processing. We use *Jess project* to classify fuzzy instances in order to eliminate the instances that have a membership degree less or equal to a certain threshold. We can then query fuzzy ontologies using *Queries* plug-in of *Protégé*. We give in the following, respectively, the pretreatment (i.e. the classification) of fuzzy instances using *Jess Project*, and an example of a query definition with *Queries* plug-in of *Protégé*.

Individuals classification of a fuzzy ontology - Jess Project

```
Defclass <Name-New-Set> (is-a <Name-Fuzzy-Concept>)
```

```
If (slot-get <--Instance--> Degree-Of-Membership >= s) then (make-instance <Instance> of: <Name-New-Set>)
```

With **s** is a threshold of membership

Treatment the user request by using Queries plug-in

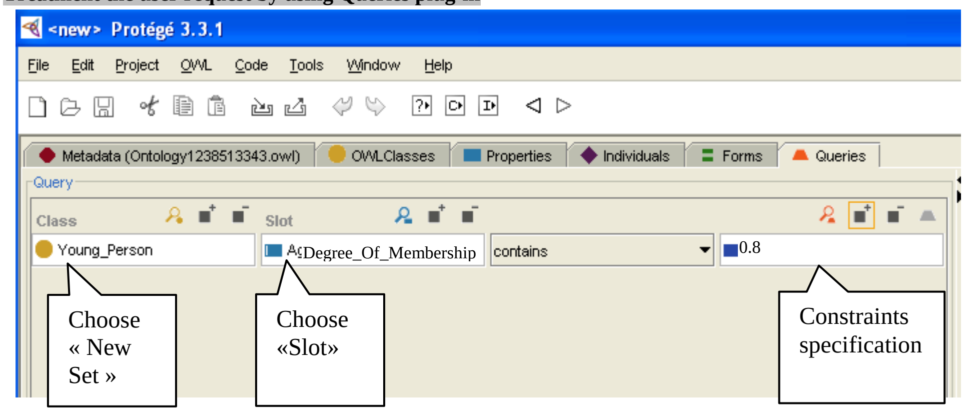

Figure.4. Fuzzy ontology classification and querying

Figure 4 presents then in the part "Treatment the user request by using Queries plug-in" an example of the requests which can be treated by Fuzzy Protégé. In this example, the user needs to find the list of the "Young Person". With this intention, he must choose the fuzzy subset (i.e. fuzzy concept) “Young_Person” in the part “Class”. Thereafter, he must also state that its search criteria are the degree of membership. He has then to put in the part “Slot” the property "Degree_Of_Membership". Finally, he must specify the minimal degree of membership to seek (in our example 0.8). With all these stages, we succeeded to treat a great range of fuzzy requests users by using the degrees of membership already calculated with the module inference and validation.

5 Conclusion and further work

In this paper, we have presented the internal architecture of *Fuzzy Protégé* and the way we can use it to implement fuzzy ontologies. *Fuzzy Protégé* aimes to be a framework which ensures the operationalization of fuzzy ontologies (building, instantiation, validating, inconsistency checking and querying). The actual version of *Fuzzy Protégé* hasbeen used for the implementation of fuzzy ontologies of "Employee's competences management" [Ghorbel, 08b]. In the future works, we project to enrich the querying module and complement the inference engine to allow inconsistency checking of fuzzy ontologies.

6 References

[Ghorbel, 08a] H. Ghorbel, A. Bahri and R. Bouaziz ; « *Les langages de description des ontologies : RDF & OWL* » ; Proc. of Génie électrique et informatique GEI'2008 ; pages 597-606 ; Sousse, Tunisie ; Mars 2008.

[Ghorbel, 08b] H. Ghorbel, A. Bahri and R. Bouaziz ; « *A Framework for Fuzzy Ontology Models* »; Proc. of journées Francophones sur les Ontologies JFO'2008 ; Lyon, France ; pages 21-30 ; Décembre 2008.

[Stoilos, 06] G. Stoilos, U. Straccia, G. Stamou et J. Pan, *« General concept inclusions in Fuzzy Description Logic »*, Proc. of the 17th European Conference on Artificial Intelligence (ECAI-06), 2006.

[Straccia, 06] U. Straccia, *« A Fuzzy Description Logic for the semantic Web »*, Proc. in Capturing Intelligence: Fuzzy logic and the semantic Web, Elie Sanchez ed., Elsevier, 2006.

[Zadeh, 75] L. Zadeh, « *The concept of a linguistic variable and it's application to approximate reasonning* » ; Proc. of Information Science ; Volume 4, Issue 4 ; pages 301-357 ; 1975.