Mapping the Fuzzy Semantic Model into Fuzzy Object Relational Database Model

Sabrine Jandoubi and Nadia Yacoubi-Ayadi
National School of Computer Sciences
University of Manouba
Tunis, Tunisia
Email: sabrine.jandoubi@gmail.com
nadia.yacoubi.ayadi@gmail.com

Afef Bahri
MIRACL Laboratory
High School of Computing and Multimedia
University of Sfax
Sfax, Tunisia
Email: afef.bahri@gmail.com

Salem Chakhar
Portsmouth Business School and Centre for Operational Research and Logistics
University of Portsmouth
Portsmouth, UK
Email: salem.chakhar@port.ac.uk

***Abstract*—This paper discusses the mapping of the Fuzzy Semantic Model (FSM) into a Fuzzy Object Relational database Model (FuzzORM). We designed a set of mapping rules to transform all the constructs of the FSM into the FuzzORM. A prototype supporting these rules is under development over the Object-Relational Database Management System (ORDBMS) PostgreSQL. The first results of implementation are presented in this paper.**

Keywords–Fuzzy database; Imperfect information; Mapping rule; Object relational database; Semantic modeling.

I. INTRODUCTION

The semantic data models are powerful conceptual modeling tools but lack effective implementation mechanisms. Thus, most of fuzzy semantic data models have been mapped and implemented through relational [1][2] or object-oriented [3][4] database models. This paper discusses the mapping of the FSM [5] into a FuzzORM. This solution permits to take advantages of both relational and object-oriented database models.

We designed a set of mapping rules to transform the constructs of the FSM into the FuzzORM. A prototype supporting these mapping rules is under development over the ORDBMS PostgreSQL. The first results of implementation are presented in this paper.

The rest of the paper is organized as follows. Section II briefly reviews the FSM. Section III details the mapping rules. Section IV presents the first implementation results. Section V discusses some related work. Section VI concludes the paper.

II. FUZZY SEMANTIC MODEL

In this section, we provide a very brief review of FSM [5].

A. Fuzzy Classes

Let E be the universe of discourse. A fuzzy entity e in E is a natural or artificial entity that one or several of its properties are fuzzy. At the extensional level, a fuzzy class K in E is a collection of fuzzy entities having some similar properties: $K = \{(e, \mu_K(e)) : e \in E \wedge \mu_K(e) > 0\}$, where $\mu_K : E \rightarrow [0,1]$ is the *membership function* that maps the elements of E to the range $[0,1]$, and $\mu_K(e)$ represents the *degree of membership* (d.o.m) of the fuzzy entity e in class K. At the intensional level, a fuzzy class K is defined as a set of attributes and a set of decision rules.

Figure 1 shows a FSM-based model example.

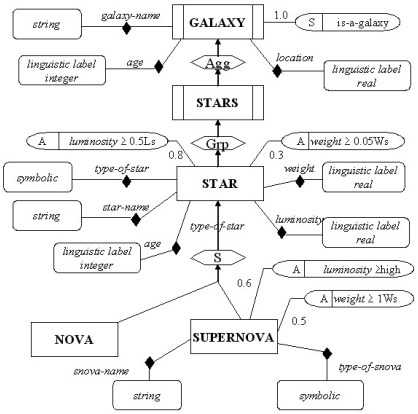

Figure 1. Example of FSM model.

For example, the class STAR in Figure 1 has five attributes (*star-name*, *type-of-star*, *age*, *luminosity* and *weight*) and two decision rules ('*luminosity*≥ 0.5' and '*weight*≥ 0.05').

B. Attributes

Each attribute is basically characterized by its name, data type and domain. A data type may be crisp or fuzzy. The domain of an attribute *attr* is the set of values the attribute may take. Let $T(attr)$ denotes the domain of *attr*. Domains of fuzzy attributes are also fuzzy. For instance, the fuzzy attribute *location* associated with class GALAXY in Figure 1 has the following domain: {in, near, very near, distant, very distant}.

C. Decision Rules

Decision rules may be based on attributes or on common semantics. An attribute-based decision rule is a condition of the form $\langle attr \rangle \langle op \rangle \langle v \rangle$, where $attr$ is an attribute, op is a binary or a set operator; and $v \in T(attr)$. A semantic decision rule is a semantic phrase used to specify the members of a fuzzy class. Two decision rules from Figure 1 are: '*luminosity*=very high' and 'is-a galaxy'. The first decision rule is based on attribute *luminosity* and associated with class STAR. The second is a semantic decision rule associated with class GALAXY. An advanced definition of decision rules is given in [6].

D. Complex Fuzzy Classes

The FSM contains several complex fuzzy classes permitting to implement the semantics of real-world in terms of generalization, specialization, aggregation, grouping and composition relationships, which are commonly used in semantic modeling. A detailed description of these constructs is given in [5].

E. Computing the Degree of Membership

The degree to which each decision rule determines the fuzzy class K is not the same. To ensure this, each decision rule j is associated with a non-negative weight w_j reflecting its importance in deciding whether or not an entity e is a member of a given fuzzy class K. The d.o.m of entity e in fuzzy class K is computed as follows [5]: $\mu_K(e) = \sum_{j=1}^{n} \rho_j(v) \cdot w_j / \sum_{j=1}^{n} w_j$, where n is the number of decision rules, $v \in D(attr)$ and $\rho_j : D(attr) \mapsto [0,1]$ is the *partial membership function* associated with the jth decision rule; it maps the elements of $D(attr)$ into $[0,1]$ (*attr* is the attribute on which the decision rule is based). For semantic decision rules, v is a semantic phrase and the partial d.o.m $\rho_j(v)$ is supposed to be equal to 1 but the user may explicitly provide a value less than 1. This basic definition of the d.o.m is used to define the membership degrees of complex fuzzy classes (see [5] for details).

III. Mapping FSM to FuzzORM Database Model

Let M be a model based on FSM. The objective of the mapping process is to create a FuzzORM database model T that captures all the semantics of M. The mapping process consists in a set of *mapping rules* to be applied on the attributes, decision rules, simple and complex classes, and semantic relationships.

A. Mapping of Attributes

Attributes in FSM can be crisp or fuzzy. The list of fuzzy data types supported by FSM are given in Table I (see also [7]). A crisp attribute is basically characterized by its name, description, domain and data type. There are other system attributes but only these ones are considered in this paper. In addition to its basic data type, a fuzzy attribute is characterized by a set of parameters permitting to generate its possibility distribution. The number of parameters needed to define fuzzy attributes is different from one data type to another.

TABLE I. Fuzzy data types.

Type	*Name*	*Example*
1	Single scalar	*quality*= average
2	Simple number	*age*=30
3	Set of possible scalar assignments	*quality*={bad,average,good}
4	Set of possible numeric assignments	*age*={20,21,22,23}
5	Fuzzy range	*age*=between 20 and 30
6	Approximate value	*age*=about 35
7	Interval	*age*$\in [25, 35]$
8	Less/More than value	*age*=less/more than 35
9	Poss. dist. over a numeric domain	*age*={0.5/20,1.0/21,0.7/22,0.3,23}
10-13	Linguistic label (four models)	*age*=young
14	An unknown value	*age*=unk
15	An undefined value	*age*=und
16	A Null value	*age*=null

Figure 2 provides the graphical representation of the possibility distribution of three examples of fuzzy attributes. The Fuzzy Range that handles 'more or less' information between two numeric values requires four parameters (α, β, γ and λ). The Approximate Value is defined by three parameters (c, c^- and c^+). The Interval data type is defined through the limits of the range α and β.

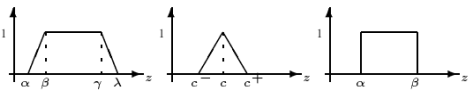

Figure 2. Graphical representation of some fuzzy data types.

In FuzzORM, the fuzzy attributes are mapped into composed and/or multi-valued attributes (supported by object relational database models) that store all the required parameters. This solution is formalized through several mapping rules. The following are two examples.

Mapping Rule 1. Let *attr* be a crisp attribute in M. Attribute *attr* is mapped to a new attribute with the same characteristics as in conventional databases.$\diamond$

Mapping Rule 2. Let *attr* be a fuzzy attribute in M. The attribute *attr* is mapped to a new attribute with the same characteristics plus an additional compound attribute *Parameters* with the following components: (i) *Value*, which is the value of the attribute as provided by the user; (ii) *DataType*, which is the fuzzy data type of the attribute provided by the user; and (iii) *ParametersList*, which is a multi-valued attribute indicating the list of parameters' values needed to generate the possibility distribution of the fuzzy data type.$\diamond$

B. Mapping of Decision Rules

The characteristics of each decision rule should be mapped into the metadata level of FuzzORM. For attribute-based decision rules, we need to maintain the following information: (i) *RuleID* that stores the identifier of the decision rule; (ii) *RelationID* that indicates the name of the class to which the decision rule is associated; (iii) *DecisionRule*, which is a composite attribute defined as follows: (a) *AttrID* that references the attribute on which the decision rule is based, (b) *Operator* that contains a binary or a set operator, and (c) *RHO*, which is a crisp or fuzzy value from the attribute domain representing the Right-Hand Operand of the decision rule; and (iv) *Weight* that stores the weight of the decision rule as specified in the data model M.

The semantic decision rules are mapped similarly. However, in this case the attribute *DecisionRule* contains two components: (a) *Operator*, which is any semantic operator such as 'IS-A', 'A-SET-OF' or 'A-PART'; and (b) *RHO*, which is a semantic phrase.

The mapping of decision rules from M to T is formalized as follows.

Mapping Rule 3. Let r_j be an attribute-based or semantic-based decision rule in the data model M. Then, the characteristics of decision rule r_j are mapped into the database model T as indicated above.$\diamond$

C. Transformation of Basic Fuzzy Classes

Each fuzzy class in the FSM model is mapped into a relation. The fuzzy attributes are mapped as explained in Section III-A. The crisp attributes are treated as in conventional databases. An additional non printable attribute, *DOM*, used

to store the global d.o.m is systematically added to the new relation. The decision rules associated with K are mapped as indicated in Section III-B.

The mapping of simple fuzzy classes is formalized as follows.

Mapping Rule 4. Let K be a fuzzy class in M with the attributes *attr*1,$\cdots$,*attrp*. The mapping of K from M to T is as follows: (i) create a relation R=(*attr*1',$\cdots$,*attrp*') where *attr*1',$\cdots$,*attrp*' are the mapping of attributes *attr*1,$\cdots$,*attrp* using Mapping Rules 1 and 2; (ii) add an attribute *DOM* with real data type to R; (iii) add the characterisers of each decision rule of K to T using Mapping Rule 3; and (iv) associate to the new relation R the triggers permitting to compte the d.o.m and to control the parameters of fuzzy attributes.$\diamond$

The last operation will be discussed in Section IV.

D. Transformation of Subclass/Superclass Relationships

A fuzzy subclass B of a fuzzy superclass A in FSM is mapped in FuzzORM into a fuzzy relation that inherits all attributes of the fuzzy relation issued from A. In addition to the attribute *DOM*, the relation B contains a new attribute, denoted by *DOM-A*, which is used to store the d.o.m of one entity from fuzzy subclass B in its fuzzy superclass A. A fuzzy subclass in FSM may be attribute-defined, roster-defined or set-operation-defined. An attribute-defined fuzzy subclass has one or several attribute values that are in accordance with some discriminative values that characterize perfectly its members. For instance, the fuzzy subclasses NOVA and SUPERNOVA in Figure 1 are specializations of the fuzzy class STAR based on the attribute *type-of-star*. A roster-defined fuzzy subclass is simply defined by an explicit enumeration of its members. A set-operation-defined fuzzy subclass may be defined as the set-difference or the set-intersection of two or more fuzzy classes.

The following mapping rule formalizes the mapping of attribute-defined subclass/superclass relationships.

Mapping Rule 5. Let B be a fuzzy subclass of a fuzzy superclass A. The mapping of subclass/superclass relationship from M to T is as follows: (i) fuzzy classes A and B are transformed according to Mapping Rule 4; (ii) a new attribute, denoted by *DOM-A* used to store the d.o.m of one entity from fuzzy subclass B in its fuzzy superclass A is added to the relation mapped from B; and (iii) add to the database model T the definition parameters of subclass/superclass relationship (i.e., list of attributes used to categorize the elements of fuzzy class B).$\diamond$

Similar mapping rules have been defined to transform roster-defined and set-operation-defined fuzzy subclasses. The main changes concerns the last operation. For roster-defined subclasses, the parameters are simply the list of the members of the fuzzy class B as specified by the user. The mapping of set-operation-defined fuzzy subclasses is more complicated since the fuzzy subclass has at least two fuzzy superclasses. Hence, the mapping rule above should be applied to each of these fuzzy superclasses. We need also to maintain the set operator used to define the subclass/superclass relationship.

E. Transformation of Interaction Relationships

An interaction relationship relates members of one fuzzy class to other members of one or many fuzzy classes. Let $B_1, \cdots, B_n$ be n fuzzy classes related by an n-ary interaction relationship. Each participant fuzzy class has $n-1$ attributes for relating each of its members to each of the other members. When a participant fuzzy class B_i is mapped into a relation in the database level, a composite attribute *InteractionList* is added to it. The *InteractionList* contains as many component attributes as the number of participant fuzzy classes. These component attributes are used to indicate the list of the related members from the other fuzzy classes.

On the other hand, when an interaction relationship requires the creation of new attributes, a new fuzzy interaction class is generated. In addition to its own attributes (that are specified in the interaction relationship), the new interaction class should contain the following attributes: (i) the key attributes of the classes participating in the interaction relationship; and a *DOM* attribute as for basic classes.

The mapping of interaction relationships is formalized as follows.

Mapping Rule 6. Let $B_1, \cdots, B_n$ be n fuzzy classes related by an n-ary interaction relationship in M. The interaction relationship is mapped as follows: (i) fuzzy classes $B_1, \cdots, B_n$ are mapped into relations $R1, \cdots, R_n$ according to Mapping Rule 4; (ii) add to each relation R_i issued from fuzzy class B_i a composite attribute *InteractionList* for relating each of its members to the members of the other classes; and (iii) if the interaction relationship requires the creation of new attributes, then a new interaction relation is created as indicated above.$\diamond$

A fuzzy class may participate in several relationships. In this case, the mapping of this fuzzy class requires as many composite attributes *InteractionList* as the number of interaction relationships.

F. Transformation of Fuzzy Complex Classes

As mentioned above, FSM supports several complex fuzzy classes (composite, aggregate or grouping classes). These classes are first mapped according to Mapping Rule 4. Then, we need to add to the metadata repository the characteristics of the semantic relationships. For instance, the composition relationships are characterized by the following information: (i) *RelationName1* that stores the name of the first fuzzy class; (ii) *RelationName2* that stores the name of the second fuzzy class; (iii) *DefinitionType* that indicates the way the composition relationship is defined (attribute-defined or enumerated); (iv) *Parameters*, which is a multi-valued attribute that stores the parameters associated with *DefinitionType* attribute (list of attributes or members); and (v) *DOM* that stores the d.o.m of the fuzzy relation named *RelationName1* in the fuzzy relation named *RealtionName2*.

A collection of other mapping rules have been defined to transform the different fuzzy complex classes but they are not presented here. However, some examples are given in Section IV-B.

IV. Implementation

A prototype named Fuzzy Interface Module (FIM) supporting the different mapping rules is under development over the ORBMS PostgreSQL. The mapping of a FSM model into FuzzORM concerns three different levels: (i) *database system level*, which is associated with extended data manipulation languages devoted to handle different fuzzy operations that the database system should support; (ii) *database level*, which is

 ISBN: 978-1-61208-386-5

concerned with the way the imperfect information is internally stored. This concerns both attribute values and extensional definition of different relations; and (iii) *metadata level*, which concerns the intensional definition of relations. The metadata level is normally managed by the host database system. However, the full implementation of the FuzzORM model requires some additional metadata, which are not supported by current database systems. In what follows, we focus on the second and third levels only.

A. Database Level

The mapping rules detailed in Section III are traduced into a set of PL/SQL routines. The main routine takes a FSM model as a text file and generates a new FuzzORM database using the different mapping rules. The input text file is scanned three times. In the first scan, FIM identifies and implements the fuzzy domains, the decision rules, the metadata about the attributes and the metadata about the semantic relationships in the FSM model. In the second scan, FIM maps all the fuzzy classes (simple or complex) and implements them without considering the semantic relationships between the classes. In the third scan, FIM adds the semantic relationships.

The fuzzy classes are mapped into relations as detailed in Section III. The mapping of the FSM model in Figure 1 leads to the FuzzORM database given in Figure 3.

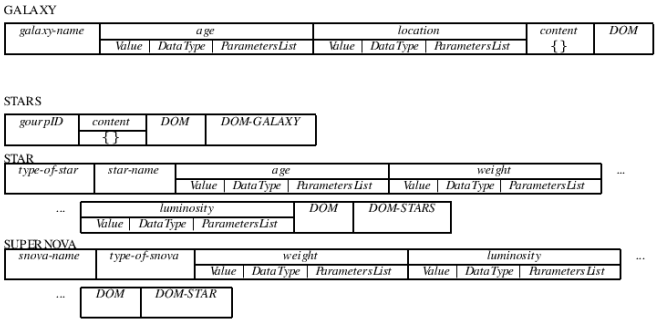

Figure 3. The FuzzORM Database.

As we can see, each fuzzy class is mapped into a relation and each fuzzy attribute into an attribute with composite type. The multi-valued attribute *content* is used in some complex fuzzy classes to maintain the list of members of these classes. This attribute is empty during the creation of the relations.

Each fuzzy relation is associated with serval triggers to control the validity of the introduced data (i.e., the parameters of fuzzy attributes). Figure 4 provides a trigger example.

```
CREATE FUNCTION GalaxyLoc() RETURNS trigger AS $GalaxyLoc$
DECLARE
 alpha numeric; beta numeric; gamma numeric; delta numeric;
BEGIN
 IF NEW.location IS NULL THEN
    RAISE EXCEPTION 'location cannot be null';
 END IF;
 IF NOT ((alpha<=beta) AND (beta<=gamma) AND (gamma<=lambda))
 THEN
   RAISE EXCEPTION 'Please provide valid parameters';
 END IF;
END;
$GalaxyLoc$ LANGUAGE plpgsql;
CREATE TRIGGER GalaxyLoc BEFORE INSERT OR UPDATE ON Galaxy
  FOR EACH ROW EXECUTE PROCEDURE GalaxyLoc();
```

Figure 4. A trigger example.

The code in Figure 4 represents a simple version of the trigger associated with the relation issued from fuzzy class GALAXY.

B. Metadata Level

The metadata contains several meta-relations for storing the different parameters and elements of FuzzORM such as fuzzy attribute characteristics, decision rules and semantic relationships. We define a meta-relation named FUZ-ATTRIBUTES to store the information about fuzzy attributes: their basic type, fuzzy types and their parameters. Here, we use a single column to define the parameters of fuzzy attributes, independently of the number of used parameters (1, 2, 3 or 4). In fact, the ORDBMS allows the use of multi-valued attributes permitting to maintain several values for a given attribute. This is not allowed in relational database systems where we should use 4 different columns to define the parameters of different fuzzy data types. This may cause many NULL values (when the number of parameters is less than 4). An extract from FUZ-ATTRIBUTES is given in Table II.

TABLE II. META-RELATION FUZ-ATTRIBUTES.

AttrID	*AttrName*	*FuzzyDataType*	*BasciDataType*	*Parameters*
1	luminosity	Fuzzy Range	Real	{0.05,0.2,1.0,1.3,}
2	weight	Interval	Real	{1.2,1.5}
3	location	Linguistic Label	Text	{7.5,10}

The meta-relation LABELS is used to handle the characteristics of linguistic labels. An extract from the meta-relation LABELS is given in Table III.

TABLE III. META-RELATION LABELS.

LabelID	*AttrID*	*Label*	*Parameters*
1	7	very young	{0,1.8}
2	7	young	{1.5,5.0}
3	7	old	{4.2,11.3}
4	7	very old	{11,15}

The labels shown in this meta-relation are relative to the fuzzy attribute *age* (which is defined as a set of Gaussian linguistic labels) associated with fuzzy class STAR.

We also define other meta-relations for handling proximity relations, possibility distributions, domains of attributes, etc.

The metadata level contains also all the information required to define the decision rules associated with different fuzzy classes. They are stored in two meta-relations: A-DECISION-RULES and S-DECISION-RULES. The first one is devoted to store the definition of attribute-based decision rules and the second one is used to store the definition of semantic decision rules. The extensional definition of the meta-relations A-DECISION-RULES and S-DECISION-RULES for our example are given in Table IV and V, respectively.

There are also a set of meta-relations to handle the semantic relationships of subclass/superclass, composition, aggregation and grouping. The first meta-relation is SUB-SUPER-COMP that is devoted to store information concerning fuzzy subclass/superclass and composition relationships. The second meta-relation GROUPING is devoted to store information concerning grouping relationships. The meta-relations SUB-SUPER-COMP and GROUPING are given in Table VI and Table VII, respectively.

 ISBN: 978-1-61208-386-5

TABLE IV. META-RELATION A-DECISION-RULES.

RuleID	*RelationID*	*RuleDefinition* *AttrID| Operator|RHO*	*Weight*
1	STAR	{luminosity,≥,$0.5L_s$}	0.80
2	STAR	{weight,≥,$0.05W_s$}	0.30
3	SUPERNOVA	{luminosity,≥,high}	0.60
4	SUPERNOVA	{weight,≥,$1W_s$}	0.50
5	NOVA	{luminosity,≥,$0.5L_s$}	0.80
5	NOVA	{weight,≥,$0.05W_s$}	0.30

TABLE V. META-RELATION S-DECISION-RULES.

RuleID	*RelationID*	*RuleDefinition* OperatorID | RHO	*Weight*
1	GALAXY	{IS-A,Galaxy}	1.0

TABLE VI. META-RELATION SUB-SUPER-COMP.

Relation1Name	*Relation2Name*	*RelationshipType*	*DefinitionType*	*Parameters*	*DOM*
GALAXY	PLANETS	Aggregation	Enumerated	{}	1.0
GALAXY	STARS	Aggregation	Enumerated	{}	1.0
GALAXY	COMETS	Aggregation	Enumerated	{}	1.0
STAR	SUPERNOVA	Subclass/ Superclass	Attribute	{star-type}	0.9
STAR	NOVA	Subclass/ Superclass	Attribute	{star-type}	1.0

TABLE VII. META-RELATION GROUPING.

RelationName	*DefinitionType*	*Parameters*	*DOM*
STARS	Enumerated	{}	1.0

The multi-valued attribute *Parameters* is used to maintain the list of members of complex fuzzy classes defined by enumeration. The list of members will be specified by the user progressively during the exploitation of the database.

C. Fuzzy Operations

To compute the membership degrees and for query processing, we need to extend the binary and the set operators that may be used in the definition of decision rules. An attribute-based decision rule is associated with a condition of the form:

$$\langle attr \rangle \langle op \rangle \langle v \rangle$$

The operator *op* may be a binary operator (i.e., $=$, $\simeq$, $\leq$, $<$, $\geq$, $>$) or a set operator (i.e., $\in$, $\subseteq$, $\subset$, $\supseteq$, $\supset$). All these operators may be associated with the negation operator, denoted '$\neg$' below. In conventional logic, the response to a binary comparison is a two-valued one and may be true (1) or false (0). Within fuzzy logic, the result of a comparison may take any value in the range [0,1]. Thus, the two-valued logic is simply a special case of fuzzy logic that is restricted to the two extreme values (0 and 1) of the range [0,1].

Based on the work of [8], we propose an extension of all the operators mentioned above to support fuzzy logic. For instance, the fuzzy operator '$\simeq$', which gives the degree in which two fuzzy numbers (approximate values in Table 1) are approximately equal is defined as follows:

$$\mu_{\simeq}(\tilde{x}, \tilde{y}) = \begin{cases} 0, & \mid \tilde{x} - \tilde{y} \mid > \text{margin}; \\ 1 - \frac{|\tilde{x}-\tilde{y}|}{margin}, & \mid \tilde{x} - \tilde{y} \mid \leq \text{margin}. \end{cases}$$

Here, we suppose that the parameters c^+ and c^- of an approximate value (see Figure 2) are the same and equal to $margin$. The fuzzy '$\neg \simeq$' operator is computed as the complement of '$\simeq$' operator, i.e., $\mu_{\neg\simeq} = 1 - \mu_{\simeq}(\tilde{x}, \tilde{y})$.

D. DOM Computing Routines

FIM proposes a set of routines for computing membership degrees. These routines are defined as triggers and associated with the corresponding relations. The basic routine named DOM is used to compute global and partial membership degrees associated to the fuzzy instances of classes (tuples in database). This routine permits to compute $\mu_K(\cdot)$ (see section II-E) and is associated to the INSERT and UPDATE triggers. The principle of DOM routine is given in Figure 5.

Algorithm 1: DOM

```
Input  : R, // relation.
         t, // tuple.
Output: real∈ [0, 1]// d.o.m.
P ⟵ {set of decision rules for R};
W ⟵ {decision rules weights};
wdom ⟵ 0;
wsum ⟵ 0;
for (each r ∈ P) do
    wdom ⟵ wdom + W[i] * RHO(r, t);
    wsum ⟵ wsum + wdom;
if (wsum = 0) then
    return 0;
return wdom/wsum;
```

Figure 5. The d.o.m computing algorithm.

The DOM routine uses the RHO function, which permits to calculate the partial membership degrees (see Section II-E).

V. DISCUSSION AND RELATED WORK

In this section, we discuss some implementation issues and some related work.

A. Discussion

The first implementation issue to discuss concerns the mapping and storing of fuzzy attributes' parameters. There are several solutions to map fuzzy attributes. We can, for example, use one common meta-relation with four attributes devoted to store the different parameters. In that time, we may have 'null' values any time the number of parameters is less than four. Another solution is to group data types along the number of required parameters. After that, four relations are needed for data types with one, two, three or four parameters, respectively. An ameliorated version of this solution is adopted in [8] where a common meta-relation is defined with a specific attribute serves as a pointer to two other meta-relations. One drawback of the solutions cited above is that anytime we need to add a new linguistic data type or to change the adopted linguistic data type, we may have to update the meta-relations structure.

The straightforward solution proposed in this paper does not depend on the parameters number because it uses multi-valued attributes allowing the storage of more than one single value. That is, all the needed parameters of linguistic data type may be defined using only one attribute. This solution has several advantages: (i) is supported by object relational database models; (ii) reduces the presence of 'null' values; and (iii) the atomic attributes of a composite attribute remain accessible individually as with non-composite attributes.

The second issue concerns the use of the attribute *Parameters* both at the intensional and extensional levels. This allows users to insert values of different data types, which may have different number of parameters. For instance, the formal definition of the attribute may be a trapezoidal-based

possibility distribution with four parameters but the user may introduce a crisp value (with no parameter at all), an interval (with two parameters) or an approximate value (with three parameters). In all cases, the different data types defined at the extensional level should be consistent with the formal definition of the attribute at the intensional level.

The third issue is related to the computing of the d.o.m. In FSM model, we need to compute the partial and global membership degrees associated with the class instances. At the database level, we need to compute the partial and global membership degrees of any tuple. Three solutions may be adopted: (i) store the partial membership degrees and compute global membership degree on the fly; (ii) store the global membership degree and compute partial membership degrees on the fly; and (iii) store both partial and global membership degrees. The first solution is expensive in access time because we need to compute the global membership degree frequently. The third solution may ameliorate the access time substantially but this needs, however, a much more storage space. In this paper, we use the second solution as it is less expensive and ameliorate access time substantially. Equally, the second approach where we compute global membership degree on preprocessing allows us to use global membership degrees as a filter to eliminate quickly false alarms in the querying process. This may reduce execution time, especially for large databases.

B. Related Work

Fuzzy information has been extensively investigated in the context of relational database model [1][2]. There are also several semantic [5][1][9][2][10][11][12] and object-oriented [13][14] database models where fuzziness is introduced with one or several levels. Due to the lack of effective implementation mechanisms, most of fuzzy semantic data models have been mapped and implemented through relational database models. For instance, the well-known Entity-Relationship (ER) data model is extended to support fuzziness in [1]. The paper includes also a fuzzy entity-relationship methodology for the design and development of fuzzy relational databases. This methodology was used for mapping the fuzzy ER to a relational one. In [15], the Is-a relationships, Functional relationships, complex Objects (IFO) model was extended to the Extended IFO (ExIFO) to represent uncertainty as well as precise information. The authors provide also an algorithm for mapping the schema of the ExIFO model to an extended NF^2 database model. In [2], the IFO data model is extended to support fuzziness. The obtained model, denoted IF_2O, is then mapped to a relational fuzzy database schema.

There are also some proposals for mapping semantic data models into object-oriented database models, which permit to support several concepts of semantic modeling. For instance, in [4] the authors extend the IFO model for handling ill-defined values including values with semantic representation, values with semantic representation and conjunctive meaning, values with semantic representation and disjunctive meaning. The paper includes also a mapping of the obtained fuzzy IFO model to a fuzzy object-oriented database model. The proposal of [3] presents an extension of the Extended Entity-Relationship (EER) model to deal with fuzzy information. The paper provides also a formal design methodology for fuzzy object-oriented databases from the fuzzy EER model.

VI. Conclusion

This paper provides a formal approach for mapping the FSM to a FuzzORM database model. Compared to existing proposals, our mapping approach permits to gather advantages of both relational and object-oriented database models. Several points related to this proposal need to be addressed: (i) the enrichment of the semantics of FSM through the addition of different constraints associated with attributes, entities and classes; and then their incorporation into FuzzORM; (ii) the definition and implementation of a data definition language adapted to FuzzORM databases; (iii) the implementation of the conceptual query language introduced in [5]; and (iv) the enrichment of FIM with capabilities related to the definition of imperative aspects of the database such as security, integrity and access levels.

References

[1] N. Chaudhry, J. Moyne, and E. Rundensteiner, "Extended database design methodology for uncertain data management," Information sciences, vol. 121, no. 1, 1999, pp. 83–112.

[2] Z. Ma, "A conceptual design methodology for fuzzy relational databases," Journal of Database Management, vol. 16, no. 2, 2005, pp. 66–83.

[3] Z. Ma, W. Zhang, W. Ma, and G. Chen, "Conceptual design of fuzzy object-oriented databases using extended entity-relationship model," International Journal of Intelligent Systems, vol. 16, no. 6, 2001, pp. 697–711.

[4] M. Vila, J. Cubero, J. Medina, and O. Pons, "A conceptual approach for dealing with imprecision and uncertainty in object-based data models," International Journal of Intelligent Systems, vol. 11, no. 10, 1996, pp. 791–806.

[5] R. Bouaziz, S. Chakhar, V. Mousseau, S. Ram, and A. Telmoudi, "Database design and querying within the fuzzy semantic model," Information Sciences, vol. 177, no. 21, 2007, pp. 4598–4620.

[6] L. Ellouze, R. Bouaziz, and S. Chakhar, "Extending fuzzy semantic model by advanced decision rules," in Annual Meeting of the North American Fuzzy Information Processing Society, 2009. NAFIPS 2009, June 2009, pp. 1–6.

[7] A. Bahri, S. Chakhar, Y. Naja, and R. Bouaziz, "Implementing imperfect information in fuzzy databases," in Proceedings of The International Symposium on Computational Intelligence and Intelligent Informatics, October 14-16 2005, pp. 1–8.

[8] J. Medina, M. Vila, J. Cubero, and O. Pons, "Towards the implementation of a generalized fuzzy relational database model," Fuzzy Sets and Systems, vol. 75, no. 3, 1995, pp. 273–289.

[9] G. Chen and E. Kerre, "Extending ER/EER concepts towards fuzzy conceptual data modeling," in Fuzzy Systems Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on, vol. 2, May 1998, pp. 1320–1325.

[10] Z. Ma, F. Zhang, L. Yan, and J. Cheng, "Fuzzy data models and formal descriptions," Studies in Fuzziness and Soft Computing, vol. 306, 2014, pp. 33–60.

[11] L. Yan and Z. Ma, "Incorporating fuzzy information into the formal mapping from web data model to extended entity-relationship model," Integrated Computer-Aided Engineering, vol. 19, no. 4, 2012, pp. 313–330.

[12] F. Zhang, Z. Ma, and L. Yan, "Representation and reasoning of fuzzy ER models with description logic DLR," Journal of Intelligent and Fuzzy Systems, vol. 26, no. 2, 2014, pp. 611–623.

[13] Z. Ma, W. Zhang, and W. Ma, "Extending object-oriented databases for fuzzy information modeling," Information Systems, vol. 29, no. 5, 2004, pp. 421–435.

[14] C. Cuevas, N. Marin, O. Pons, and M. Vila, "Pg4DB: A fuzzy object-relational system," Fuzzy Sets Systems, vol. 159, no. 12, Jun. 2008, pp. 1500–1514.

[15] A. Yazici, B. Buckles, and F. Petry, "Handling complex and uncertain information in the ExIFO and NF^2 data models," IEEE Transactions on Fuzzy Systems, vol. 7, no. 6, 1999, pp. 659–676.

 ISBN: 978-1-61208-386-5