Conceptual Design and Implementation of the Fuzzy Semantic Model

Chakhar Salem† and Abdelkader Telmoudi‡
†LAMSADE laboratory, University of Paris Dauphine,
Place du Maréchal de Lattre de Tassigny, 75775 Paris Cedex 16, Paris, France.
Email: chakhar@lamsade.dauphine.fr
‡Department of Industrial Engineering, National School of Engineering of Tunis,
B.P. 37 Le Belvédère 1002 - Tunis, Tunisia.
Email: abdelkader.telmoudi@enit.rnu.tn

***Abstract*— FSM is one of few database models that support fuzziness, uncertainty and impreciseness of real-world at the class definition level. FSM authorizes an entity to be partially member of its class according to a given degree of membership that reflects the level to which the entity verifies the extent properties of this class. This paper deals with the conceptual design of FSM and adresses some implementation issues.**

***Index Terms*— Database design, Fuzzy database, Database mapping, Imperfect information.**

I. Introduction

In database literature, we enumerate several extensions of semantic and object-oriented database models to support the fuzziness, uncertainty and impreciseness of real-world [1], [5], [10], [12], [13]. Most of these extensions introduce fuzziness only at the attribute level and consider that entities are fully encapsulated into their classes, which means that they fully verify the properties of these classes.

There are, however, some proposals for extending object-oriented [7], [8], [11], [15] and semantic [6], [14] database models to support fuzziness, uncertainty and impreciseness of real-world at the class definition level. In the same direction of research, the authors have proposed a new data model, namely fuzzy semantic model (FSM) [2], [4], that authorizes an entity to be partially member of its class according to a given degree of membership that reflects the level to which the entity verifies the extent properties of this class.

This paper deals with the conceptual design of FSM and adresses some implementation issues. Section II introduces briefly FSM. Section III details the conceptual design of classes and subclasses in FSM. Section IV discuses some implementation issues. Section V concludes the paper.

II. Fuzzy Semantic Model

A. Basic idea

The *space of entities* E is the set of all entities of the interest domain. A *fuzzy entity* e in E is a natural or artificial entity that one or several of its properties are fuzzy. In other words, a fuzzy entity verifies only (partially) some extent properties of its class. A *fuzzy class* K in E is a collection of fuzzy entities: $K = \{(e, \mu_K(e)) : e \in E \wedge \mu_K(e) > 0\}$. μ_K is a characteristic or *membership function* and $\mu_K(e)$ represents the *degree of membership* (d.o.m) of the fuzzy entity e in the fuzzy class K. Membership function μ_K maps the elements of E to the range $[0, 1]$ where 0 implies no-membership and 1 implies full membership. A value between 0 and 1 indicates the extent to which entity e can be considered as an element of fuzzy class K.

B. Entity/Class membership functions in FSM

A fuzzy class is a collection of fuzzy entities having some similar properties. Fuzziness is thus induced whenever an entity verifies only (partially) some of these properties. We denote by $X_K = \{p_1, p_2, ..., p_n\}$ (with $n \geq 1$) the set of these properties for a given fuzzy class K. X_K is called the *extent* of fuzzy class K. The extent properties may be derived from the attributes of the class and/or from common semantics. The degree to which each of the extent properties determines fuzzy class K is not the same. Indeed, there are some properties that are more discriminative than others. To ensure this, we associate to each extent property p_i a non-negative weight w_i reflecting its importance in deciding whether or not an entity e is a member of a given fuzzy class K. We also impose that $\sum_{i=1}^{n} w_i > 0$.

On the other hand, an entity may verify fully or partially the extent properties of a given fuzzy class. Let D^i be the basic domain of extent property p_i values and P^i is a subset of D^i, which represents the set of possible values of property p_i. The *partial membership function* of an extent property value is $\rho_{P_K^i}$ which maps elements of D^i into $[0, 1]$. For any attribute value $v_i \in D^i$, $\rho_{P_K^i}(v_i) = 0$ means that fuzzy entity e violates property p_i and $\rho_{P_K^i}(v_i) = 1$ means that this entity verifies fully the property. The number v_i is the value of the attribute of entity e on which the property p_i is defined. For extent properties based on common semantics, v_i is a semantic phrase and the partial d.o.m $\rho_{P_K^i}(v_i)$ is supposed to be equal to 1 but the user may explicitly provide a value less than 1. More generally, the value of $\rho_{P_K^i}(v_i)$ represents the extent to which entity e verifies property p_i of fuzzy class K. Thus, the *global* d.o.m of the fuzzy entity e in the fuzzy class K is:

$$\mu_K(e) = \frac{\sum_{i=1}^{n} \rho_{P_K^i}(v_i) \cdot w_i}{\sum_{i=1}^{n} w_i}$$

C. Constructs of FSM

In FSM each fuzzy class is uniquely identified with a name. Each class has a list of characteristics or properties, called attributes. Some of these attributes are used to construct the extent set X_K defined above. To be a member of a fuzzy class K, a fuzzy entity e must verify (fully or partially) at least one of the extent properties, i.e., $\mu_K(e) > 0$.

The classes in FSM are categorized as exact or fuzzy:

- An *exact class* K is a class that all its members have a d.o.m equal to 1; i.e., $\mu_K(e) = 1 \forall e \in K$.
- A *fuzzy class* K is a class that at least one of its members has a d.o.m strictly inferior to 1; i.e., $\exists e \in K$ such that $\mu_K(e) < 1$.

Classes may also be categorized as strong or weak:

- A *strong fuzzy class* is a fuzzy class whose members can exist on their own, i.e., they are not depending on other classes.
- A *weak fuzzy class* is a fuzzy class whose members depend on the existence of other (strong or weak) classes for their existence.

These two classifications are orthogonal and all combinations are possible.

The elements of a fuzzy class are called *members*. In FSM, α-MEMBERS denotes for a given fuzzy class K the set $\{e : e \in K \wedge \mu_K(e) \geq \alpha\}$; where $\alpha \in [0, 1]$. It is easy to see that α-MEMBERS $\subseteq$ β-MEMBERS for all α and β in [0,1] and verifying $\alpha \geq \beta$. Note that 1-MEMBERS may also be refereed to *true* or *exact members*. In turn, α-MEMBERS with $0 < \alpha < 1$ are called *fuzzy members*.

The concept of α-MEMBERS may be mapped to the concept of α-cut associated with fuzzy sets and which is defined for a fuzzy subset F as the set $F_\alpha = \{x : \mu_F(x) \geq \alpha\}$ with $0 \leq \alpha \leq 1$.

FSM supports four different relationships: property, decision-rule, membering and interaction. Property relationships relate fuzzy classes to domain classes. Each property relationship creates an attribute and each attribute has a unique datatype and may be single-valued, unknown, undefined, null or multi-valued. Decision rule relationships are implementation of the extents of fuzzy classes, i.e., the set of properties-based rules used to assign fuzzy entities to fuzzy classes. Membering relationships relate fuzzy entities to fuzzy classes through the definition of d.o.m. Interaction relationships relate members of one fuzzy class to other members of one or several fuzzy classes.

In FSM there are several complex fuzzy classes (see Table I), that permit to implement the semantics of real-world among objects in terms of generalization, specialization, aggregation, grouping and composition relationships, which are commonly used in purely semantic modelling.

To make the paper short, the ways to define membership functions at the interaction, subclass/superclass relationships, composition, aggregation and grouping, and the more general class/class relationships are not addressed here. They are detailed in [2], [3].

TABLE I
FSM COMPLEX FUZZY CLASSES

Class	*Description*
Interaction fuzzy class	A fuzzy class that describe the interaction of two or more fuzzy classes
Fuzzy superclass	A generalization of one or many, simple or complex, fuzzy classes
Fuzzy subclass	A specialization of one or many, simple or complex, fuzzy classes
Composite fuzzy class	A fuzzy class that has other fuzzy classes as members
Aggregate fuzzy class	A fuzzy class that its members are heterogeneous and exhaustive collection from several fuzzy classes
Grouping fuzzy class	A fuzzy class that its members are homogenous collection of members from the same fuzzy class

III. Schema definition in FSM

This section provides a proposal for specifying schema of FSM-based databases. All examples of this section rely on the database example of Figure 1. In the example database, GALAXY is an aggregate fuzzy class whose members are unique collections of members from COMETS, STARS and PLANETS fuzzy grouping classes. These last ones are homogenous collections of members from strong fuzzy classes COMET, STAR and PLANET, respectively. NOVA and SUPERNOVA are two attribute-defined fuzzy subclasses of STAR basing on *type-of-star* attribute. PLANET-TYPES is an attribute-defined fuzzy composite class. This composition is from PLANET fuzzy class basing on the *age* attribute. PERSON is an exact class. It has three enumerated subclasses: SCIENTIST, TECHNICIAN and OFFICER. Each person is affiliated with at least one LABORATORY. SCIENTIST is a collection of scientists and DISCOVERY is an interaction fuzzy class between SUPERNOVA and SCIENTIST. SCIENTIST-TYPES is a fuzzy composite class from SCIENTIST basing on *field-of-research* attribute.

In the generic definitions below we have adopted the following conventions:

- [ ] : optional parameter(s).
- { } : list of parameters or values.
- | : the equivalent of the binary operator “xor”.
- < > : obligatory parameter(s).
- ( ) : series of parameters connected with the “xor” operator, i.e. only one of the parameters delimited with “(” and “)” is chosen.

The generic definition of a fuzzy class in FSM is as follows:

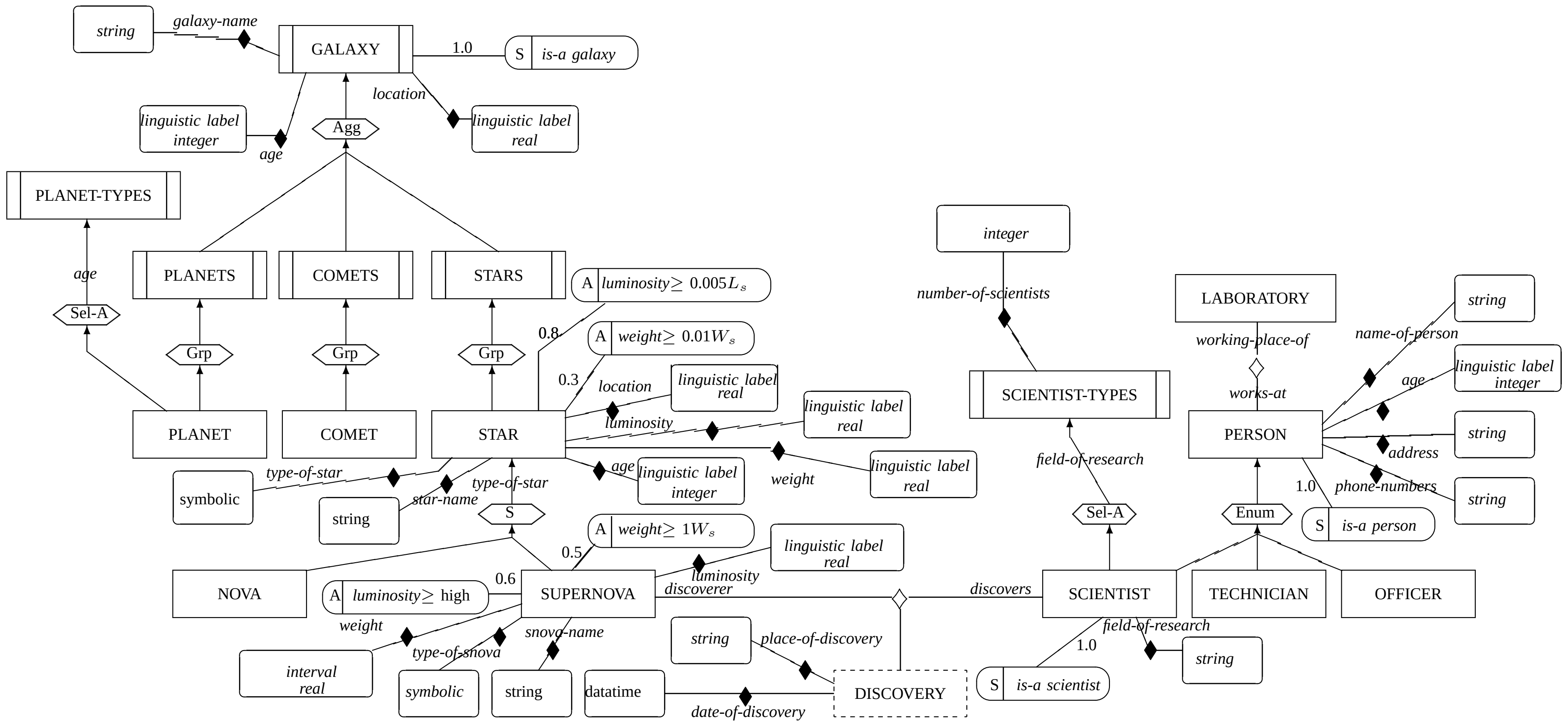


Fig. 1. Example of a FSM-based model

CLASS <*class-name*> WITH DOM OF <*dom*>
{
SUPERCLASS:
OF <*sclass-name-1*> WITH DOM OF <*dom-1*>
...
OF <*sclass-name-k*> WITH DOM OF <*dom-k*>

INTERACTION CLASS OF <*class-list-1*>

EXTENT:
<*ext-pr-1*> WITH WEIGHT OF <w_1> DECISION RULE IS ((<*attr-name-1*><*op*> (<*s-attr-name-1*>| <*value*>))|<*op*> <*sphrase-1*>)
...
<*ext-pr-n*> WITH WEIGHT OF <w_n> DECISION RULE IS ((<*attr-name-n*><*op*> (<*s-attr-name-n*>| <*value*>)) | <*op*><*sphrase-n*>)

ATTRIBUTES:
<*attr-name-1*>: [FUZZY] DOMAIN <*domaine-1*>: TYPE OF <*type-1*> WITH DOM OF <*dom-1*>: [REQUIRED][UNIQUE] [MULTI-VALUED]
...
<*attr-name-r*>: [FUZZY] DOMAIN <*domaine-r*>: TYPE OF <*type-r*> WITH DOM OF <*dom-r*>: [REQUIRED][UNIQUE] [MULTI-VALUED]

CONTENTS:
[ENUMERATED COMPOSITION FROM (<*class-name-1:members-list-1*>)]
[SELECTED COMPOSITION ON ATTRIBUTES <*attr-list-1*> FROM <*class-list-2*>]
[AGGREGATION OF (<*class-name-2:members-list-2*>)]
[GROUPING FROM <*class-name-3:members-list-3*>]

INTERACTION:
<*inter-name-1*> WITH (<*class-name-1*> INVERSE IS <*inv-inter-name-1*> | <*inter-class-list-1*>) [CLASS IS <*inter-class-name-1*>]
...
<*inter-name-z*> WITH (<*class-name-z*> INVERSE IS <*inv-inter-name-z*> | <*inter-class-list-z*>) [CLASS IS <*inter-class-name-z*>]
}

The SUPERCLASS component of the fuzzy class definition enumerates all the subclasses of the class along with their d.o.m relatively to this class. This component is omitted if the fuzzy class has no fuzzy subclass(es). The INTERACTION CLASS OF component is for fuzzy interaction classes only. It permits to specify the list of the participant classes for which the interaction class is defined. One, two and at least three class names are required for recursive, binary, and n-ary ($n \geq 3$) fuzzy interaction classes, respectively. Next in the EXTENT part, we list all the extent properties of the class (We remak that fuzzy interaction classes have no EXTENT component since they have no extent properties.) For each extent property we indicate the name, the weight and the decision rule on which this extent property is based. As it is quoted earlier, decision rules may be attribute-based or semantic phrase-based. The left-side of the attribute-based rule indicates the attribute name on which the rule is based. The op operator may be a scalar comparator (e.g. $=$,$\neq$,$<$,$>$,$\leq$,$\geq$) or a set-operator (e.g. $\in$, $\subset$, $\subseteq$). The extension of all these operators to operate on imperfect information is provided in section IV.D. The right-side of the attribute-based may be a crisp (e.g. *age*=21) or fuzzy (e.g. *age*=young) value. For semantic phrase-based decision rules, the op is an "is-a" operator and the right-side is a semantic phrase (e.g. the decision rule "is-a person" may be associated with the class PERSON in Figure 1). The semantic phrase-based rules are optional—but recommended to make the database schema more comprehensible. For instance, we may have the following extent properties definitions:

p_1 WITH WEIGHT OF 0.8 DECISION RULE IS *luminosity* = very high
p_2 WITH WEIGHT OF 0.3 DECISION RULE IS *weight* in $[0.01W_s - 1W_s]$
p_3 WITH WEIGHT OF 0.5 DECISION RULE IS *age*= young
p'_3 WITH WEIGHT OF 0.5 DECISION RULE IS *age* in [17-21]
p_4 WITH WEIGHT OF 1.0 DECISION RULE IS is-a galaxy
p_5 WITH WEIGHT OF 1.0 DECISION RULE IS is-a person

The symbol "W_s" above is the weight of the sun; it is often used as a measurement unit. The four first decision rules are attribute-based ones while the last two decision rules are semantic phrase-based ones. Decision rules p_4 and p_5 may be associated with classes GALAXY and PERSON, respectively.
In the ATTRIBUTES component we specify the list of the attributes of the fuzzy class. We note that attributes definition is partially inspired from [11]. This definition of attributes apply for both exact and fuzzy ones. An exact attribute requires the definition of a datatype (e.g. integer, string) and a domain as a range of possible values for the attribute. A fuzzy attribute requires the definition of a fuzzy type and a fuzzy domain. The fuzzy types are based on simple (e.g. integer) or complex types (e.g. set-valued types, entity-valued attributes) that allow the representation of imprecise information. Fuzzy domains may be represented simply as a list of fuzzy linguistic terms (e.g. young, near). Other ways may also apply as for example possibility theory (e.g. the age of a young person may be represented through a possibility distribution as *age* = $0.1/17 + 1.0/18 + 0.2/19$) or evidence theory (e.g. through evidence theory, the age of young person is *age* = $0.1/\{18\} + 0.1/\{18, 19\} + 0.8/\{17, 18, 19\}$). In addition, attributes may be specified as required, unique or multi-valued (Note: if the MULTI-VALUED keyword is not specified, the attribute is a single-valued one.) Required attributes are those that must have non-null values and unique attributes are those for which no two members of the same class may have the same value. All required and unique attributes may serve as *identifiers* (or *keys*) that mean to identify all the members of a fuzzy class. For example, we may have the following declarations of attributes:

location: FUZZY DOMAIN {*in, near, very near, distant, very distant*}: TYPE OF *real* WITH DOM OF 1.0
age: FUZZY DOMAIN {*very old, old, young, very young*}: TYPE OF *integer* WITH DOM OF 1.0
star-name: TYPE OF *string* WITH DOM OF 1.0: REQUIRED UNIQUE
phone-numbers: TYPE OF *string* WITH DOM OF 1.0: MULTI-VALUED

According to these declarations, *location* and *age* attributes may have either exact values (e.g. *location*= 12LY; *age*=55) or fuzzy values (e.g. *location*=very distant; *age*=old). The “LY” symbol is the abbreviation of light year. The *star-name* attribute may have only exact values (e.g. *star-name*= Vega). In addition, *star-name* attribute may be used as an identifier since it is required and unique. The *phone-numbers* attribute is an exact and multi-valued one.

The next component of fuzzy class definition is specific for fuzzy composite and grouping classes. For enumerated composition, we indicate the list of classes and for each one we specify the entities that are member of the fuzzy composite class. For attribute-based composition we fix the list of the selection attributes and the list of the classes from which selection is accomplished. For fuzzy aggregated classes we indicate the list of the classes that are part of the aggregation and for each one we specify the entities that are member of the fuzzy aggregate class. And finally for fuzzy grouping classes, we indicate the name of the class from which grouping is realized along with the list of members.

The last part of fuzzy class definition indicates the eventual interaction relationship(s) of the fuzzy class. As mentioned earlier, interaction relationships may be binary or n-ary. In both cases a name should be provided. Binary interaction relationships require also the name of the other participating fuzzy class and the name of the inverse attribute. For n-ary interaction relationships we need to mention the list of the classes that participate in this interaction. In both cases and when it is necessary, the name of the fuzzy interaction class can be specified with the CLASS IS component.

Since subclasses may have their own subclasses, they have the same components as for fuzzy classes. In particular, they may have SUPERCLASS components that indicate the list of their own subclasses. In turn, subclasses have a specific component, called SPECIALIZATION, that is designed to map to their fuzzy superclasses. The generic definition of a fuzzy subclass in FSM is as follows (in this definition, only the SPECIALIZATION component is provided; the definitions of the other components is similar to the ones of the fuzzy class and they are not reproduced):

SUBCLASS <*sclass-name*> WITH DOM OF <*dom*>
{
SPECIALIZATION :
OF <*class-name-1*> WITH DOM OF <*dom-1*>:
[BY ENUMERATION <*members-list-1*>]
[ON ATTRIBUTES <*attr-list-1*>]
[BY INTERSECTION WITH <*class-list-1*>]
[BY DIFFERENCE WITH <*d-class-name-1*>]

OF <*class-name-q*> WITH DOM OF <*dom-q*>:
. . .
}

For each superclass of the subclass, we indicate the name of the superclass and the d.o.m of the subclass in this superclass. A subclass may be defined in four ways. Enumerated fuzzy subclasses require the enumeration of the fuzzy classes that participate in the generalization relationship along with the list of members. For attribute-defined subclasses, we should indicate the list of the attributes on which the ISA relationship is defined. For set-intersection-defined subclasses we indicate simply the list of the other superclasses that participate in the intersection. Finally for difference-defined subclasses we mention the name of the other fuzzy class that participate in the difference operation.

To better illustrate these definitions, we provide in the following several examples based on Figure 1.

CLASS *galaxy* WITH DOM OF *gdom*
{
EXTENT:
gp_1 WITH WEIGHT OF 1.0 DECISION RULE IS set of galaxies

ATTRIBUTES:
galaxy-name: TYPE OF *string* WITH DOM OF 1.0: REQUIRED UNIQUE
age: FUZZY DOMAIN {*very young, young, old, very old*}: TYPE OF *integer* WITH DOM OF 1.0: REQUIRED
location: FUZZY DOMAIN {*in, near, very near, distant, very distant*}: TYPE OF *real* WITH DOM OF 1.0: REQUIRED

CONTENTS:
AGGREGATION OF *comets:* c_1, c_2, c_3, *stars:* s_1, s_2, *planets:* p_1, p_2, p_3, p_4, p_5
}

CLASS *star* WITH DOM OF *sdom*
{
SUPERCLASS:
OF *supernova* WITH DOM OF *scdom-sn*
OF *novae* WITH DOM *scdom-n*

EXTENT:
sp_1 WITH WEIGHT OF 0.8 DECISION RULE IS $luminosity \geq 0.005L_s$
sp_2 WITH WEIGHT OF 0.3 DECISION RULE IS $weight \geq 0.05W_s$

ATTRIBUTES:
star-name: TYPE OF *string* WITH DOM OF 1.0: REQUIRED UNIQUE
type-of-star: TYPE OF *symbolic*(*nova, supernova*) WITH DOM OF 1.0: REQUIRED
age: FUZZY DOMAIN {*very young, young, old, very old*}: TYPE OF *integer* WITH DOM OF 1.0: REQUIRED
location: FUZZY DOMAIN {*in, near, very near, distant, very distant*}: TYPE OF *real* WITH DOM OF 1.0: REQUIRED
luminosity: FUZZY DOMAIN {*very low, low, medium, high, very high*}: TYPE OF *real* WITH DOM OF 1.0: REQUIRED
weight: FUZZY DOMAIN $[0.01Ws - 100W_s]$: TYPE OF *real* WITH DOM OF 1.0: REQUIRED
}

SUBCLASS *supernova* WITH DOM OF *sndom*
{
SPECIALIZATION :
OF *star* WITH DOM OF *scdom*:
ON ATTRIBUTES *type-of-star*

EXTENT:
snp_1 WITH WEIGHT OF 0.6 DECISION RULE IS *luminosity* $\geq$ high
snp_2 WITH WEIGHT OF 0.5 DECISION RULE IS *weight* $\geq 1W_s$

ATTRIBUTES:
snova-name: TYPE OF *string* WITH DOM OF 1.0: REQUIRED UNIQUE
type-of-snova: TYPE OF *symbolic*(*Ia, Ib, Ic, Ib/c, Ic/b, II-P, II-L*) WITH DOM OF 1.0: REQUIRED
luminosity: FUZZY DOMAIN {*high, very high*}: TYPE OF *real* WITH DOM OF 1.0: REQUIRED
weight: FUZZY DOMAIN $[1W_s - 100W_s]$: TYPE OF *real* WITH DOM OF 1.0: REQUIRED

INTERACTION:
discoverer WITH *scientist* INVERSE IS *discovers* CLASS IS *discovery*
}

CLASS *person* WITH DOM OF 1.0
{
SUPERCLASS:
OF *scientist* WITH DOM OF 1.0
OF *technician* WITH DOM OF 1.0
OF *officer* WITH DOM OF 1.0

EXTENT:
pp_1 WITH WEIGHT OF 1.0 DECISION RULE IS set of persons

ATTRIBUTES:
name-of-person: TYPE OF *string* WITH DOM OF 1.0: REQUIRED
age: FUZZY DOMAIN {*very young, young, old, very old*}: TYPE OF integer WITH DOM OF 1.0: REQUIRED
address: TYPE OF *string* WITH DOM OF 1.0: REQUIRED
phone-numbers: TYPE OF *string* WITH DOM OF 1.0: MULTI-VALUED

INTERACTION:
works-at WITH *laboratory* INVERSE IS *working-place-of*
}

CLASS *discovery* WITH DOM OF *sndom*
{
INTERACTION CLASS OF *supernova, scientist*

ATTRIBUTES:
date-of-discovery: TYPE OF *datetime* WITH DOM OF 1.0
place-of-discovery: TYPE OF *string* WITH DOM OF 1.0
}

SUBCLASS *scientist* WITH DOM OF 1.0
{
SPECIALIZATION:
OF *person* WITH DOM OF 1.0:
BY ENUMERATION *name-of-person-1,name-of-person-2,name-of-person-3,name-of-person-4*

EXTENT:
scp_1 WITH WEIGHT OF 1.0 DECISION RULE IS set of scientists

ATTRIBUTES:
field-of-research: TYPE OF *string* WITH DOM OF 1.0: REQUIRED

INTERACTION:
discovers WITH *supernova* INVERSE IS *discoverer* CLASS IS *discovery*
}

IV. Implementation issues

This section first shows how different kinds of imperfect information are represented and internally implemented. It then provides a formal approach to map FSM-based model to a fuzzy relational object (FRO) database model. Finally, it shows how scalar and set-operators should be extended to operate on imperfect information.

A. Imperfect information representation

FRO supportes a rich set of imperfect data types that are listed below. Note that these data types are extensions of the ones proposed in [9]. We also added several new ones. Especially, linguistic labels defined on sinusoidal possibility distributions and the “more than” and “less than” data types are not defined in [9].

- *Fuzzy range*. This data type handles the “more or less” information between two numeric values. The graphical representation of possibility distribution of this data type is shown through Model I.1 in Table II and may be written as $\{\mu(z)/z : z \in D\}$. D is the domain of the attribute values and $\mu(z)$ is the d.o.m of z in the fuzzy set on which the attribute is defined. This set is denoted A in Table II. As it is shown in Table II, four parameters are required to define the possibility distribution of this data type: α, β, γ and λ. The parameters β and γ represent the support of the fuzzy set associated with the attribute values and α and λ represent the limits of the transition zones;
- *Approximate value*. This data type handles the “about” some numeric value information. The graphical representation of possibility distribution of this data type is shown through Model I.2 in Table II and may be written as $\{\mu(z)/z : z \in D\}$. Here, three parameters are required: the central value of the concept c, the limit of left transition zone c^- and the limit of right transition zone c^+;
- *Interval*. Model I.3 in Table II shows the graphical representation of the possibility distribution of a classical crisp range. Mathematically, this possibility distribution may be written as $\{\mu(z)/z : z \in D\}$. The parameters required here are the limits of the range α and β;
- *Less/More than value*. These data types focalize only on one side of a value. The graphical representations of the possibility distributions of “less than” and “more than” data types are shown in Models I.4 and I.5 in Table II, respectively. Mathematically, the possibility distribution associated with both of them may be written as $\{\mu(z)/z : z \in D\}$. Two parameters are required to define this data type: the value of interest (γ or β) and the limit of the transition range (λ or α);
- *Set of possible scalar assignments*. This permits to handle attributes defined on a set of scalars. For example, the *height* of a person may be defined as *height*={tall,very tall}, which is represented through possibility distribution

as {1.0/tall,1.0/very tall}. A proximity relation is often defined on the domain of this data type. We denote this data type with Model III.1;

- *Set of possible numeric assignments*. This data type is similar to the previous one. It differs only on the fact that is defined on a set of numeric values. For example, the *height* of a person may be defined as the set {1.85,1.95}, which is represented through possibility distribution as {1.0/1.85,1.0/1.95}. This data type will be designed as Model III.2;
- *Possibility distribution over discrete domain*. This data type is represented through standard possibility distribution where possibility degrees in [0,1] are associated with each of the domain values. More formally, we have $\{p_1/d_1, \cdots, p_n/d_n\}$; where p_i and d_i for i trough 1 to n are the possibility degrees and the domain values, respectively. Note that the domain values may be numbers as well as scalars. A proximity relation is often associated with scalar-based domains. This data type will be designed as Model III.3;
- *Possibility distribution over a numeric ordered domain*. In this data type, the possibility distribution is defined on an ordered set of numeric values as for example *age*={0.7/25,0.8/26,1.0/27,0.8/28,0.8/30}. More generally, we have $\{p_1/d_1, \cdots, p_n/d_n\}$ with $p_i \leq p_{i+1}$. This data type will be designed as Model III.4.
- *Simple number*. This is a crisp data type which is handled as in conventional databases. The possibility distribution-based representation of a simple number n is $\{1.0/n\}$. Model I.8 in Table II shows the graphical representation of the possibility distribution of this data type;
- *Simple scalar*. This is a crisp data which is handled as in conventional databases. The possibility distribution-based representation of a simple scalar s is {1.0/s}. A proximity relation is often associated with the domain of this data type. We denote this data type with Model III.5;
- *Matching degree*. This is a real number in [0,1] that refers to the degree to which a concept is achieved (e.g. *quality*=0.7). The possibility distribution-based representation of a matching degree m is $\{1.0/m\}$. This data type will be designed as Model III.6;
- *Unknown*. This data type means that we cannot decide which is the value of the attribute among several plausible values. But the attribute may take any value from its domain. Accordingly, the possibility distribution-based representation of the unknown data type is $\{1.0/z : z \in D\}$. Model I.6 in Table II shows the graphical representation of the possibility distribution of this data type;
- *Undefined*. This data type means that there is not any defined value that can be assigned to the attribute. This means that no one of the domain values is authorized. Accordingly, the possibility distribution-based representation of undefined data type is $\{0/z : z \in D\}$. Model I.7 in Table II shows the graphical representation of the possibility distribution of this data type;
- *Null*. This data type means that we cannot even know whether the attribute's value is unknown or undefined. Accordingly, the possibility distribution-based representation of undefined data type is $\{1.0/$Unknown,$1.0/$Undefined$\}$. This data type will be designed as Model III.7;
- *Symbolic*. This is a crisp data type which takes its values on a set of symbolic values related with the XOR operator. For instance, the attribute *type-of-star* associated with the class STAR in Figure 3 may be only *nova* or *supernova*. The possibility representation of this data type is $\{0/s_1, \cdots, 1.0/s_i, \cdots, 0/s_r\}$ which means that the attribute value is s_i. This data type will be designed as Model III.8;
- *Linguistic label*. Models II.1-II.4 in Table II are the graphical representation of the possibility distribution of the linguistic label data types. Model I.1 represents the sinusoidal model. The parameters required here are the central value of the attribute c and the parameter that governs the shape of the d.o.m a. Model II.2 is an extension of the previous one that applies when the central value of the concept may take a range of values instead of only one value. Four parameters are required here: the limits of the central range a_1 and a_2; and the left and right transition zones b_1 and b_2, respectively. Note that a_1 and a_2 are the *crossover* (or *transition*) *points* defined such that $\mu(a_1) = \mu(a_2) = 0.5$. Models II.3 and II.4 are the asymmetric extensions of Model II.1 that apply when only the left or right side of the concept is of interest. The required parameters are a_1 and b_1 for Model II.3; and a_2 and b_2 for Model II.4. The mathematical representation of all these data types is $\{\mu(z)/z : z \in D\}$. Finally, note that proximity relations need to be associated with the domains of these four data types.

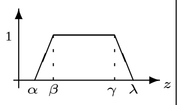
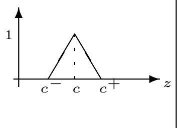
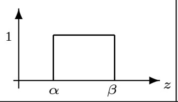
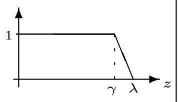
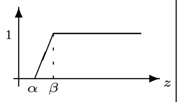
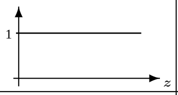
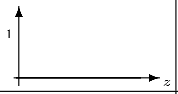
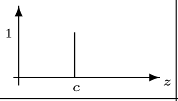
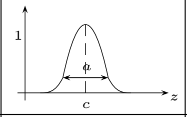
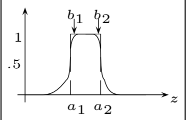
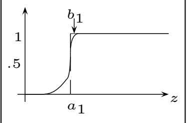
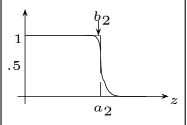
B. Imperfect information implementation

In order to store the specificity of all the attributes, we define a meta-relation, called ATTRIBUTES, at the metadata level with the following attributes:

- *attribute-id*: it uniquely identifies each attribute defined at the database level. It constitutes also the primary key of the ATTRIBUTES meta-relation. Note that the key attribute(s) in this relation and in the other ones are underlined.
- *attribute-name*: it stores the name of an attribute. As for classical databases, the same fuzzy class can not have two attributes with the same name but the same attribute name may appear in different fuzzy classes.
- *class-name*: denotes the fuzzy class to which the attribute belongs.
- *data-type*: which is a multi-valued attribute that stores the attribute type which may take any one of the list of §IV.A. For crisp attributes, this attribute works as in conventional databases (it may take the values of integer, real, float, etc.). For fuzzy attributes, the *data-type* attribute stores the fuzzy data type itself and the basic crisp data type on which the fuzzy data type is based.

TABLE II

DIFFERENT DATA TYPES SUPPORTED BY FSS

Data type A^1	Model	Representation	Parameters	$\mu_A(z)$
Fuzzy range label e.g. *age*= more or less between 20 and 30	I.1		$\alpha, \beta, \gamma, \lambda$	$\mu_A(z) = \begin{cases} 1, & \text{if } \beta \le z \le \gamma; \\ \frac{\lambda - z}{\lambda - \gamma}, & \text{if } \gamma < z < \lambda; \\ \frac{z-\alpha}{\beta - \alpha}, & \text{if } \alpha < z < \beta; \\ 0, & \text{Otherwise.} \end{cases}$
Approximate value e.g. *age*=about 35	I.2		c, c^-, c^+	$\mu_A(z) = \begin{cases} 1, & \text{if } z = c; \\ \frac{c^+ - z}{c^+ - c}, & c < z < c^+; \\ \frac{z - c^-}{c - c^-}, & c^- < z < c; \\ 0, & \text{Otherwise.} \end{cases}$
Interval e.g. *age* $\in [25, 35]$	I.3		α, β	$\mu_A(z) = \begin{cases} 1, & \text{if } \alpha \le z \le \beta; \\ 0, & \text{Otherwise.} \end{cases}$
Less than value e.g. age= less than 35	I.4		γ, λ	$\mu_A(z) = \begin{cases} 1, & \text{if } z \le \gamma; \\ 0, & \text{if } z < \lambda; \\ \frac{\lambda - z}{\lambda - \gamma}, & \text{if } \gamma \le z \le \lambda. \end{cases}$
More than value e.g. age= more than 35	I.5		α, β	$\mu_A(z) = \begin{cases} 1, & \text{if } z \ge \beta; \\ 0, & \text{if } z \le \alpha; \\ \frac{z-\alpha}{\beta-\alpha}, & \text{if } \alpha < z < \beta. \end{cases}$
Unknown	I.6			$\mu_A(z) = 1 \; ; z \ge 0$
Undefined	I.7			$\mu_A(z) = 0 \; ; z \ge 0$
Real number e.g. *age*=30	I.8		c	$\mu_A(z) = \begin{cases} 1, & \text{if } z = c; \\ 0, & \text{Otherwise.} \end{cases}$
Linguistic label e.g. *age*=young	II.1		a, c	$\mu_A(z) = \frac{1}{(1+(a(z-c)^2} ; z \ge 0$
Linguistic label e.g. *age*=young	II.2		a_1, a_2, b_1, b_2	$\mu_A(z) = \begin{cases} \frac{1}{1+\frac{z-a_1-b_1}{b_1}}, & \text{if } z < a_1 + b_1; \\ 1, & \text{if } a_1 + b_1 \le z \le a_2 - b_2; \\ \frac{1}{1+\frac{z-a_2+b_2}{b_2}}, & \text{if } z > a_2 - b_2. \end{cases}$
Linguistic label *age*=very old	II.3		a_1, b_1	$\mu_A(z) = \begin{cases} \frac{1}{1+\frac{z-a_1-b_1}{b_1}}, & \text{if } z < a_1 + b_1; \\ 1, & \text{if } a_1 + b_1 \le z; \end{cases}$
Linguistic label e.g. *age*=very young	II.4		a_2, b_2	$\mu_A(z) = \begin{cases} 1, & \text{if } z \le a_2 - b_2; \\ \frac{1}{1+\frac{z-a_2+b_2}{b_2}}, & \text{if } z > a_2 - b_2. \end{cases}$

The ATTRIBUTES meta-relation associated with the model in Figure 1 is as follows (only some attributes are shown):

attribute-id	*attribute-name*	class-name	data-type
attr-15	*star-name*	STAR	{string}
attr-16	*type-of-star*	STAR	{symbolic}
attr-17	*age*	STAR	{linguistic label, real}
attr-18	*luminosity*	STAR	{linguistic label, real}
attr-19	*location*	STAR	{linguistic label, real}
attr-20	*weight*	STAR	{interval, real}
attr-60	*name-of-person*	PERSON	{string}
attr-61	*address*	PERSON	{string}
attr-62	*phone-numbers*	PERSON	{string}
attr-77	*field-of-research*	SCIENTIST	{scalar}
attr-80	*age*	PLANET	{linguistic label, real}

The parameters associated with different linguistic terms that appear in the domain of any linguistic data type are stored at the metadata level. They will be used to compute the different d.o.m and for query processing. The number of parameters needed is different from one linguistic data type to another and it may vary from zero to four parameters. Thus, several solutions are possible to store these parameters. 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 associated with one linguistic value 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 (we do not have to define a relation for unknown and undefined attribute data and other data types that need no parameter). An ameliorated version of this solution is adopted in [9]. The authors use a common meta-relation similar to ATTRIBUTES and a specific attribute serves as a pointer to two meta-relations. One meta-relation is used to store the "margin" parameter needed for approximately data type (Model I.2 in Table II). The second meta-relation contains a list of fuzzy objects defined in the database columns. This meta-relation contains two specific attributes: one used to store the data type and the other points out to three new meta-relations devoted to store the parameters of qualifier labels defined over the matching of a query, proximity relations associated with scalar data types (Models II.1-II.4 in Table II and Model III.5 in the list of §IV.A), and trapezoidal-based possibility distribution (Models I.1-I3 in Table II) of linguistic labels and query quantifiers, respectively. In the last meta-relation, four attributes (*Alpha*, *Beta*, *Gamma*, *Delta*) are used to store the trapezoidal-based possibility distributions parameters. In the special case of interval data type, the attributes *Alpha* and *Beta* store the same value. This is also true for attributes *Gamma* and *Delta*. The same meta-relation with the four parameters is also used to store undefined, unknown and null data types, which generates an excessive storage space since these data types require no parameters and the different parameters will be "null"-valued.

One drawback of the solutions cited above is that any time we need to add a new linguistic data type or to change the adopted linguistic data types, we may have to update the meta-relations structures. Here, we propose a straightforward solution that does not depend on the parameters number and can be used with any fuzzy model. In fact, we define a common meta-relation with a multi-valued attribute (supported by relational object database models) that stores all needed parameters. This meta-relation, denoted by PARAMETERS, contains one line for each linguistic value that appears in the domain of any linguistic data type attribute (or the list of the authorized values for symbolic data type). Its attributes are:

- *attribute-id*: references one attribute that appears in the meta-relation ATTRIBUTES.
- *label*: stores a linguistic term belonging to the attribute domain. For symbolic data types this attribute takes a "nil" value.
- *parameters*: is a multi-valued attribute used to store the parameters required for generating the possibility distribution of the linguistic term.

Intuitively, attributes with no parameters, will not be included in PARAMETERS meta-relation.

An example of a PARAMETERS meta-relation associated with the model in Figure 1 is as follows:

attribute-id	*label*	*parameters*
attr-16	nil	{nova, supernova}
attr-17	very young	$\{0.0, 0.0, 0.5, 1\}$
attr-17	young	$\{0.8, 1.7, 2, 2.5\}$
attr-17	old	$\{2.3, 5, 10, 15\}$
attr-17	very old	$\{12, 17, 50, 60\}$

As it is shown in the ATTRIBUTES meta-relation, *attr-16* and *attr-17* correspond to the *type-of-star* and *age* attributes in fuzzy class STAR, respectively; and values on attribute *parameters* (for *attr-17*) are expressed in million of years.

The meta-relation PARAMETERS permits also to generate the domain of linguistic or symbolic data types. This needs only to group together all the linguistic labels having the same *attribute-id* in the meta-relation PARAMETERS. For example, the domain of attribute *attr-17* above is {very young, young, old, very old}. The domain of a symbolic data type is the list of the terms in the *parameters* attribute.

The metadata level contains also the information required to define the extent properties of fuzzy classes. These information are used to compute the partial and global d.o.m. They are stored in two meta-relations called A-DECISION-RULES and S-DECISION-RULES. The A-DECISION-RULES is devoted to store attribute-based extent properties. It has the following attributes:

- *extent-property*: stores the name of the extent property.
- *class-name*: denotes the name of the fuzzy class for which the extent property is defined.
- *based-on*: references the *attribute-id* on which the extent property is based.
- *decision-rule*: is a composite attribute defined as follows:
  - *operator*: contains a binary (=, *approx-equal*, $\leq$, $\geq$, $<$, $>$, $\neq$) or a set ($\subset$, $\subseteq$, $\supseteq$, $\supset$, $\in$) operator.
  - *right-hand-operand*: is a crisp or fuzzy value from the attribute domain.
- *weight*: stores the weight of the extent property.

An example of a S-DECISION-RULES meta-relation associated with the model in Figure 1 is as follows:

extent-property	*class-name*	*based-on*	*decision-rule*	*weight*
			operator|*right-hand-operand*	
ext-star-1	STAR	attr-17	$\{\geq, 0.005L_s\}$	0.8
ext-star-2	STAR	attr-18	$\{\geq, 0.5W_s\}$	0.3
ext-snova-1	SUPERNOVA	attr-50	{=, high}	0.6
ext-snova-2	SUPERNOVA	attr-51	$\{\geq, 1W_s\}$	0.5

Note that the symbols W_s and L_s are the weight and luminosity of the Sun, respectively; they are often used as

measurement units.
The S-DECISION-RULES is devoted to store extent properties based on common semantics. It has the following attributes:

- *extent-property*: stores the name of the extent property.
- *class-name*: denotes the name of the fuzzy class for which the extent property is defined.
- *decision-rule*: is a composite attribute defined as follows:
  - *operator*: is an "is-a" operator.
  - *right-hand-operand*: is a semantic phrase.
- *weight*: stores the weight of the extent property.

An example of a S-DECISION-RULES meta-relation associated with the model in Figure 3 is as follows:

extent-property	*class-name*	*decision-rule* *operator|right-hand-operand*	*weight*
ext-person	PERSON	{is-a, person}	1.0
ext-galaxy	GALAXY	{is-a, galaxy}	1.0
ext-scientist	SCIENTIST	{is-a, scientist}	1.0

The attribute values are stored at the database level. As mentioned above, to facilitate data manipulation and for computing efficiency, the different types of attributes values are uniformly represented through possibility distribution. However, these distributions are not explicitly stored in the database but generated automatically during data manipulation and query processing by means of specific functions associated with different data types.
As they are defined in [2], attributes values may be crisp, fuzzy or both. This need only to be indicated in the intent definition of the fuzzy classes the attributes belong to. FSS allows users to insert values of any data type that is consistent with the formal definition of the attribute. At the extent definition of the fuzzy class, each fuzzy attribute is mapped into a new composite one composed of three component attributes:

- *attr-value*: stores the value of the attribute as provided by the user.
- *data-type*: stores the data type of the value being inserted.
- *parameters*: is a multi-valued attribute used to store parameters associated with the attribute value that are used to generate its possibility distribution.

The *data-type* attribute is used both at the extent definition and in the intent definition to allow users insert values of different data types, which may have different number of parameters. This will offer more flexibly to the user. Nevertheless, the different data types defined at the extent level should be consistent with the formal definition of the attribute at the intent level. For instance, the formal definition of the attribute may be a trapezoidal-based possibility distribution with four paraments but the user may introduce a crisp value (with no parameter at all), an interval (with two parameters only) or an approximate value (with three parameters only). Remark that the attribute *data-type* at the extent definition is not a multi-valued one. Note also that for computing the partial d.o.m, the parameters defined at the intent level are used to define the possibility distribution of the *left-hand-operand* of the extent property.
The extent definition of fuzzy class STAR from Figure 1 will look as follows (only *luminosity* and *weight* attributes are shown):

luminosity *attr-value|data-type|parameters*	*weight* *attr-value|data-type|parameters*
{high, linguistic label model II.1, {25,5}}	{$10W_S$, real, {nil}}
{$0.1L_S$, real,{nil}}	{[$12W_S$-$15W_S$], interval, {12,15}}
{more than $10L_S$, more than linguistic label, {$7.5L_S$,10}}	{about $17W_S$, approximate value, {15,17,18}}

Some data types (Models II.1-II.4, III.1, III.3 and III.5) require also to define the proximity relation between the elements of their respective domains. Proximity relations are stored at the metadata level through the meta-relation PROXIMITY which has the following attributes:

- *attribute-id*: references the attribute for which the proximity relation is defined.
- *label-1* and *label-2*: denote two linguistic terms belonging to the attribute domain.
- *degree*: stores the similarity degree between the linguistic terms *label-1* and *label-2*.

The following is the meta-relation PROXIMITY for proximity relation of the attribute *age* associated with the fuzzy class STAR in Figure 1:

attribute-id	*label-1*	*label-2*	*degree*
attr-17	very young	young	0.7
attr-17	very young	old	0.1
attr-17	very young	very old	0.0
attr-17	young	old	0.1
attr-17	young	very old	0.0
attr-17	old	very old	0.8

Proximity relations are reflexive and symmetric. Thus, there is no need to handle the proximity degrees for pairs of the type (x, x) and only one pair from (x, y) and (y, x) should be stored for any two linguistic labels x and y.
In the case of fuzzy subclass/superclass, composition, aggregation and grouping relationships, we need to store the d.o.m of one fuzzy (sub)class in the corresponding fuzzy superclass, composite, aggregate or grouping class. To handle these information, we add two new meta-relations SUB-SUPER-COMP and GROUPING.
The meta-relation SUB-SUPER-COMP is devoted to store information concerning fuzzy subclass/superclass and composition relationships. It contains the following attributes:

- *class-1-name*: stores the name of the first fuzzy class.
- *class-2-name*: stores the name of the second fuzzy class.
- *relationship*: stores the type of relationship between the fuzzy classes denoted by *class-1-name* and *class-2-name* (i.e. subclass/superclass or composition).
- *definition-type*: indicates the way the relationship is defined (attribute-defined, roster-defined, set-intersection or set-difference for fuzzy subclass/superclass relationships; and attribute-defined or enumerated for fuzzy composition relationships).
- *parameters*: is a multi-valued that stores the parameters associated with *definition-type* attribute.
- *dom*: stores the d.o.m of the fuzzy class denoted by *class-2-name* in the fuzzy class denoted by *class-1-name*.

For attribute-based fuzzy subclass/superclass and attribute-based fuzzy composition relationships, the attribute *parameters* stores the attributes on which the generalization/specialization or the composition is based. For roster-based fuzzy subclass/superclass and enumerated fuzzy composition relationships, the attribute *parameters* takes the "nil" value.

The meta-relations SUB-SUPER-COMP associated with Figure 1 is as follows:

class-1-name	*class-2-name*	*relationship*	*definition-type*	*parameters*	*dom*
STAR	SUPERNOVA	Subclass/Superclass	Attribute-defined	{attr-17}	0.5
STAR	NOVEA	Subclass/Superclass	Attribute-defined	{attr-17}	0.7
PERSON	SCIENTIST	Subclass/Superclass	Roster-defined	{nil}	1.0
PERSON	OFFICER	Subclass/Superclass	Roster-defined	{nil}	1.0
PERSON	TECHNICIAN	Subclass/Superclass	Roster-defined	{nil}	1.0
SCIENTIST-TYPES	SCIENTIST	Composition	Attribute-defined	{attr-77}	1.0
PLANET-TYPES	PLANET	Composition	Attribute-defined	{attr-80}	0.45

The meta-relation GROUPING is devoted to store information concerning fuzzy grouping and aggregation relationships. It is similar to SUB-SUPER-COMP but it contains only *class-1-name*, *class-2-name*, *relationship* and *dom* attributes.
The meta-relation GROUPING associated with Figure 1 is as follows:

class-1-name	*class-2-name*	*relationship*	*dom*
STARS	STAR	Grouping	1.0
PLANETS	PLANET	Grouping	1.0
COMETS	COMET	Grouping	1.0
GALAXY	SATRS	Aggregation	0.5
GALAXY	PLANETS	Aggregation	0.9
GALAXY	COMETS	Aggregation	0.7

We also add a new meta-relation, called INTERACTION, devoted to store information concerning fuzzy interaction relationships. It contains the following attributes:

- *class-name*: stores the name of a fuzzy class participating in the interaction *interaction-name*.
- *interaction-name*: stores the name of the fuzzy interaction class or the name of the fuzzy interaction relationship.
- *role*: stores the name of the relationship from the point of view of *class-name*.

The meta-relation INTERACTION associated with Figure 1 is as follows:

class-name	*interaction-name*	*role*
SUPERNOVA	DISCOVERY	*discoverer*
SCIENTIST	DISCOVERY	*discovres*
LABORATORY	*working*	*working-place-of*
PERSON	*working*	*works-in*

C. Mapping FSM-based model to a fuzzy relational object database model

As mentioned earlier, FSM-based model is mapped into a fuzzy relational object (FRO) database one. Here we provide the transformation of only simple classes, Fuzzy subclass/superclass relationships and interaction relationships.

1) Fuzzy simple classes transformation: Each fuzzy class in the FSM model is mapped into a relation in the database level. The fuzzy attributes are mapped into composite ones as explained above. 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 into the new relation. In addition, the information relative to the extent properties of the fuzzy class are automatically introduced in the A-DECISION-RULES and/or S-DECISION-RULES meta-relations.
As example, the mapping of the fuzzy class STAR in Figure 1 is as follows (only a subset of the attributes are included).

star-name	*type-of-star*	*luminosity* *attr-value|data-type|parameter*	*dom*
Vega	NOVA	{low, linguistic label II.1,{2,0.5}}	0.3
3C 58	SUPERNOVA	{0.1L_S, real,{nil}}	1.0
Proxima Centauri	SUPERNOVA	{more than 10L_S, more than linguistic label,{7.5L_S,10}}	0.75

Remark particularly how the attribute *luminosity*is mapped into composite one as explained in §IV.A.

2) Fuzzy subclass/superclass relationships transformation: A fuzzy subclass B of a fuzzy superclass A is mapped into a relation which inherits all the relevant attributes of the relation transformed from A (the relational object database model allows inheritance). 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. The same reasoning is used for fuzzy subclasses with more than one fuzzy superclass. Note particulary that the relation mapped from fuzzy class B will contain several *dom-A*, one for each fuzzy superclass.
The mapping of the fuzzy subclass SUPERNOVA in Figure 1 is as follows (the attribute *discovery-list* is added to indicate for each supernova the list of its discoverers as explained in the following paragraph):

snova-name	*type-of-snova*	$\cdots$	*discovery-list* *attr-value|data-type|parameter*	*dom*	*dom-star*
SN1987a	IIb	$\cdots$	{Ian Shelton}	0.95	1.0
SN1604	Ic	$\cdots$	{Johannes Kepler}	0.5	0.5
SN1006	Unknown	$\cdots$	Unknown	0.7	0.9

3) Interaction relationships transformation: Let B_1, B_2,$\cdots$, B_n be n fuzzy classes related by an n-ary interaction relationship *inter-name*. When a participant fuzzy class B_i is mapped into a relation in the database level, a composite attribute *inter-name-list* is added to it. The *inter-name-list* contains as many component attributes as the number of participant fuzzy classes in *inter-name*. Each component attribute of fuzzy subclass B_j is used to indicate, for each member from B_i, the list of the members from B_j in interaction with the one of B_i. See for example the mapping of fuzzy subclass SUPERNOVA in Appendix B.2 where the attribute *discovery-list* is added to it to indicate for each supernova the list of its discoverers.
Note that a fuzzy class may participate in several relationships and the same reasoning apply for each interaction relationship. Especially, the mapping of this fuzzy class requires as many composite attributes *inter-name-list* as the number of interaction relationships.
On the other hand, when an interaction relationship requires the creation of new attributes, a new fuzzy interaction class is generated. When the fuzzy interaction class is mapped into a relation in the database level, it contains, in addition to its own attributes, the following ones:

- *interaction-id*: this attribute is generated by the system and used to identify interaction relationships.
- B_i*-id*: used to store the key attribute value of the member of participating fuzzy class B_i. When the interaction relationship is reflexive, two attributes are used to reference members of the same fuzzy class.
- *dom*: denotes the global d.o.m of fuzzy interaction class members.

The mapping of the fuzzy subclass DISCOVERY in Figure 1 is as follows.

interaction-id	*date-of-discovery*	*place-of-discovery*	*supernova-id*	*scientist-id*	*dom*
inter-1	1987	China	SN1987a	Ian Shelton	1.0
inter-22	1604	Netherlands	SN1604	Johannes Kepler	0.9
inter-55	1006	Unknown	SN1006	Unknown	0.75

Remark especially the attributes *date-of-discovery* and *place-of-discovery* which are specific to fuzzy subclass DISCOVERY.

D. Computation of the degrees of membership

To compute the partial and global d.o.m and for query processing, we need to extend the binary and the set operators that may be used in the definition of the extent properties of fuzzy classes. As it is detailed earlier, each attribute-based extent property is associated with a condition of the form: <*left-hand-operand*> <*op*> <*right-hand-operand*>. The operator op may be a binary operator (i.e. $=, \simeq, \leq, <, \geq, >$) or a set operator (ie. $\in, \subseteq, \subset, \supseteq, \supset$). All these operators may be associated with the negation operator "not" denoted "$\neg$" below. In conventional logic, the response to a binary comparison is a two-valued one and may be true (or 1) or false (or 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].
Basing of the work of [9], in the rest of this section we propose an extension of all the operators mentioned above.

- The fuzzy "$=$" and "$\neg =$" operators:

  This fuzzy "$=$" operator models the equality concept for precise as well as imprecise data values. Four different membership functions can be distinguished:

$\mu_{=}(X,Y)$	crisp y	fuzzy $\tilde{y}$
crisp x	$\begin{cases} 1, & x = y; \\ 0, & \text{otherwise.} \end{cases}$	$sup_{(x,y)\in XxY}\ \min[1, \pi_{\tilde{y}}(y)]$
fuzzy $\tilde{x}$	$\pi_{\tilde{x}}(y)$	$sup_{(x,y)\in XxY}\ \min[p(x,y), \pi_{\tilde{x}}(x), \pi_{\tilde{y}}(y)]$

  where $p(x,y)$ is the proximity relation, and $\pi_{\tilde{x}}$ and $\pi_{\tilde{y}}$ are the possibility distribution defined on the domains X and Y. The fuzzy "$\neg =$" operator is simply defined as $1 - \mu_{=}(\tilde{x},\tilde{y})$.

- The fuzzy "$\simeq$" and "$\neg \simeq$" operators:

  The fuzzy "$\simeq$" operator gives the degree in which two fuzzy numbers (approximate values in Table 1) are approximately equal. It is computed as follows:
  $$\mu_{\simeq}(\tilde{x},\tilde{y}) = \begin{cases} 0, & \mid \tilde{x}-\tilde{y} \mid > margin; \\ 1 - \frac{\mid \tilde{x}-\tilde{y} \mid}{margin}, & \mid \tilde{x}-\tilde{y} \mid \leq margin. \end{cases}$$
  Here we suppose that the parameters c^{+} and c^{-} of an approximate value (see §6.1 and Table 1) are the same and equal to $margin$. The fuzzy "$\neg \simeq$" operator is computed as the complement of "$\simeq$" operator, i.e. $\mu_{\neg\simeq} = 1 - \mu_{\simeq}(\tilde{x},\tilde{y})$.

- The fuzzy "$\leq$" and "$\neg \leq$" operators:

  The "$\leq$" operator is defined on ordered domains and apply for both crisp and imprecise data. Accordingly, four situations hold:

$\mu \leq (X,Y)$	crisp y	fuzzy $\tilde{y}$
crisp x	$\begin{cases} 1, & x \leq y; \\ 0, & \text{otherwise.} \end{cases}$	$\min_{y\leq x; y\in Y} \pi_{\tilde{y}}(x)$
fuzzy $\tilde{x}$	$\max_{x\leq y; y\in Y} \pi_{\tilde{x}}(y)$	$\mu \leq (X,Y)\ \sup_{(x,y)\in XxY} \min[\pi_X(x), \pi_Y(y)]$

  The "$\neg \leq$" operator is computed as the complement of "$\leq$", i.e. $\mu_{\neg\leq}(\tilde{x},\tilde{y}) = 1 - \mu_{\leq}(\tilde{x},\tilde{y})$.

- The fuzzy "$\geq$" and "$\neg \geq$" operators:

  The fuzzy "$\geq$" operator is defined on ordered domains and apply for both crisp and imprecise data. Four situations can be distinguished:

$\mu \geq (X,Y)$	crisp y	fuzzy $\tilde{y}$
crisp x	$\begin{cases} 1, & x \geq y; \\ 0, & \text{otherwise.} \end{cases}$	$\max_{y\geq x; y\in Y} \pi_{\tilde{y}}(y)$
fuzzy $\tilde{x}$	$\min_{x\leq y; y\in Y} \pi_{\tilde{y}}(y)$	$\mu \geq (X,Y)\ \sup_{(x,y)\in XxY} \min[\pi_X(x), \pi_Y(y)]$

  The fuzzy "$\neg \geq$" operator is defined as the complement of "$\geq$", i.e. $\mu_{\neg\geq} = 1 - \mu_{\geq}(\tilde{x},\tilde{y})$.

- The fuzzy "$>$" and "$\neg >$" operators:

  The fuzzy "greater than" operator "$>$" is defined as the complement of the "$\leq$" operator, i.e. $\mu_{>} = 1 - \mu_{\leq}(\tilde{x},\tilde{y})$. The fuzzy "$\neg >$ operator is defined as $\mu_{\neg>} = 1 - \mu_{>}(\tilde{x},\tilde{y})$

- The fuzzy "$<$" and "$\neg <$" operators:

  The fuzzy "less than" operator "$<$" is defined as the complement of the "$\geq$" operator, i.e. $\mu_{<} = 1 - \mu_{\geq}(\tilde{x},\tilde{y})$. It complement is defined as $\mu_{\neg>} = 1 - \mu_{<}(\tilde{x},\tilde{y})$.

- The fuzzy "$\in$" and "$\ni$" operators:

  The fuzzy "$\in$" operator permits to compute the d.o.m to which a crisp data y is in a fuzzy one $\tilde{x}$. The formula is $\mu_{\in}(\tilde{x},y) = \pi_{\tilde{x}}(y)$. The fuzzy "$\ni$" operator is simply defined as $\mu_{\ni} = 1 - \mu_{\in}(\tilde{x},y)$.

- The fuzzy "$\subseteq$", "$\neg \subseteq$", "$\supseteq$" and "$\neg \supseteq$" operators:

  The fuzzy "$\subseteq$" operator permits to compute the d.o.m to which a fuzzy data $\tilde{y}$ is included a fuzzy data $\tilde{x}$. The formula is $\mu_{\subseteq}(\tilde{x},\tilde{y}) = \min_{z\in X\cap Y}(\pi_{\tilde{x}}(z), \pi_{\tilde{y}}(z))$. The fuzzy "$\neg \subseteq$" operator is simply defined as $1 - \mu_{\subseteq}(\tilde{x},\tilde{y})$. The two fuzzy operators "$\supseteq$" and "$\neg \supseteq$" are similar to "$\subseteq$" and "$\neg \subseteq$", respectively.

- The fuzzy "$\subset$", "$\neg \subset$", "$\supset$" and "$\neg \supset$" operators:

  The fuzzy "$\subset$" measures the degree to which a fuzzy data is strictly included in another fuzzy data. It is defined as the complement of the "$\supseteq$", i.e. $\mu_{\subset} = 1 - \mu_{\supseteq}(\tilde{x},\tilde{y})$. The fuzzy "$\neg \subset$" is defined as $\mu_{\neg\subset} = 1 - \mu_{\subset}(\tilde{x},\tilde{y})$. In the same way, the fuzzy "$\supset$" operator is defined as the complement of "$\subseteq$" operator, i.e. $\mu_{\supset} = 1 - \mu_{\subseteq}(\tilde{x},\tilde{y})$.

The fuzzy "$\neg\subset$" is defined as $\mu_{\neg\supset} = 1 - \mu_{\subseteq}(\tilde{x}, \tilde{y})$.

To compute the different d.o.m, the prototype contains several functions that are activated automatically through the TRIGGERS associate with the relations. As an example, we cite the GLOBAL-DOM function that permits to compute the global d.o.m for each entity being introduced in the database. This function implements Equation presented in §II.B. It is associated with the INSERT and UPDATE triggers and has two parameters: the name of the class (class-name) and the identifier of the entity (entity-id). The class-name is used to get the extent properties of the class and their corresponding weights as well as the attributes domains and, eventually, the proximity relations associated with the different domains. The entity-id is used to get the values of the attributes of the entity being inserted or modified. The general schema of the GLOBAL-DOM function is as follows provided below.
The function GLOBAL-DOM uses the PARTIAL-DOM which permits to calculate the partial d.o.m of the different extent properties of the class. This function takes as parameters *left-hand-operand*, *op* and *right-hand-operand* and returns the partial d.o.m. Along with with the type of operator *op*, this function activates one of the extended fuzzy binary or set operators enumerated above.

```
Function GLOBAL-DOM (class-name, entity-id)
BEGIN
  P ← {set of extent properties}
  W ← {$w_i$}
  size=$|P|$
  wdom ←0
  wsum ←1
  For i=1 to size
      wdom=wdom+W[i]*PARTIAL-DOM(class-name.P[i,1],class-name.P[i,2],class-name.P[i,3])
      wsum=wsum+W[i]
  EndFor
  Return wdom/wsum
END
```

The others functions are defined in similar way but to make the paper short, their descriptions in not included here.

V. Conclusion

This paper deals with the conceptual design of FSM and adresses some implementation issues. First, we have provided a proposal for specifying FSM schemas. Then, we have addressed some issues related to the implementation of FSM. More specifically, we have showed how different kinds of imperfect information are represented and internally implemented. Then we have briefly described a formal approach to map FSM-based model to a fuzzy relational object (FRO) database model. Finally, we have given some insights concerning the extension of scalar and set-operators to operate on imperfect information.

References

[1] Bordogna, G. Pasi, and D. Lucarella. A fuzzy object-oriented data model for managing vague and uncertain information. *International Journal of Intelligent Systems*, 14:623–651, 1999.
[2] R. Bouaziz, S. Chakhar, V. Mousseau, R. Sudah, and A. Telmoudi. Database design, implementation and querying within the fuzzy semantic model. *Information Sciences*, to appear, In Press.
[3] R. Bouaziz, S. Chakhar, and I. Saad. Membership functions definition in the fuzzy semantic model. In *Proceedings of the International Conference: Sciences of Electronic, Technologies of Information and Telecommunications (SETIT'04)*, Sousse, Tunisia, March 27-31 2005.
[4] S. Chakhar and A. Telmoudi. Extending database capabilities: Fuzzy semantic model. In *Proceedings of the International Conference: Sciences of Electronic, Technologies of Information and Telecommunications (SETIT'04)*, Sousse, Tunisia, March 15-20, 2004.
[5] N. Chaudhry, J. Moyne, and E.A. Rundensteiner. An extended database design methodology for uncertain data managmenet. *Journal of Database Management*, 121(1/2):83–112, 1999.
[6] G.Q. Chen and E.E. Kerre. Extending ER/EER concepts towards fuzzy conceptual data modeling. In *Proceedings of the 1998 IEEE International Conference on Fuzzy Systems*, pages 1320–1325, 1998.
[7] R. George, R. Srikanth, F.E. Petry, and B.P. Buckles. Uncertainty management issues in object-oriented data model. *IEEE Transaction on Fuzzy Systems*, 4:179–192, 1996.
[8] N.V. Gyseghem and R.D. Caluwe. Imprecision and uncertainty in UFO database model. *Journal of American Society of Information Sciences*, 49(3):236–252, 1998.
[9] J.C. Cubero J.M. Medina, M.A. Vila and O. Pons. Towards the implementation of a generalized fuzzy relational database model. *Fuzzy Sets and Systems*, 75:273–289, 1995.
[10] Z.M. Ma. A conceptual design methodology for fuzzy relational databases. *Journal of Database Management*, 16(2):66–83, 2005.
[11] Z.M. Ma, W.J. Zhang, and W.Y. Ma. Extending object-oriented databases for fuzzy information modeling. *Information Systems*, 29:421–435, 2004.
[12] Z.M. Ma, W.J. Zhang, W.Y. Ma, and G.Q. Chen. Conceptual design of fuzzy object-oriented databases using extended entity-relationship model. *International Journal of Intelligent Systems*, 16:697–711, 2001.
[13] M.A. Vila, J.C. Cubero, J.M. Medina, and O. Pons. A conceptual aproach for deal with imprecision and uncertainty in object-based data model. *International Journal of Intelligent Systems*, 11:791–806, 1996.
[14] A. Yazici, B.P. Buckles, and F.E. Petry. Handling complex and uncertain information in the ExIFO and NF^2 data models. *IEEE Transactions on Fuzzy Systems*, 7(6):659–676, 1999.
[15] A. Yazici, R. George, and D. Aksoy. Design and implementation issues in fuzzy object-oriented data model. *Information Science*, 108:241–260, 1998.