

Towards Soft Computing Object-Oriented Logic Programming

J.F. Baldwin, T.H. Cao¹, T.P. Martin and J.M. Rossiter

AI Group

Department of Engineering Mathematics

University of Bristol

United Kingdom BS8 1TR

{Jim.Baldwin, Tru.Cao, Trevor.Martin, Jonathan.Rossiter}@bristol.ac.uk

Abstract

Logic programming, object-oriented programming and soft computing have provided advantageous methodologies and techniques for computer-based problem solving. This paper proposes a framework that combines these three disciplines to exploit their own advantages in dealing with real world problems. The framework is a logic-based one in which class and object properties are represented by clauses. Vague data in properties are represented by fuzzy sets interpreted as possibility distributions. Uncertain applicability of a property to a class or an object is represented either by a support pair defining probability lower and upper bounds, or by a certainty lower bound. Fundamental issues of uncertain membership and inheritance are then discussed and solutions to them are proposed. The result forms a basis for development of soft computing object-oriented programming systems.

1. Introduction

For dealing with real world problems, which are too complicated to be solved by a single methodology or technique, combinations of different disciplines have been the quest and focus of significant research effort. In particular, logic programming and object-oriented programming have been combined, resulting in object-oriented logic programming languages (e.g. [30], [13], [26]). At this juncture, on the one hand, logic programming provides a declarative way for problem specification, well-founded semantics for formal reasoning, and non-deterministic mechanism for solution searching. On the other hand, object-oriented programming provides class and object structures, which can be naturally mapped to problem domains, for designing systems and writing programs on a large scale.

Meanwhile, soft computing has emerged and been developed as being essential for representing and reasoning with vague and uncertain information, which is pervasive in the real world. In fact, soft computing is not a single methodology, but a partnership of fuzzy logic, probabilistic reasoning and neuro-computing ([38]) and in

this paper we consider only the first two partners. Fuzzy logic and probabilistic logic have been applied to extend both logic programs (e.g. [15], [24], [29], [32], [28]) and object-oriented models (e.g. [7], [33], [36], [14]) to deal with vagueness and uncertainty often encountered in practical problems.

However, research on combining all together logic programming, object-oriented programming and soft computing appears to be sporadic. The benefit of such a combination is not only that the advantages of the three disciplines can be exploited, but also that one discipline sheds light on existing problems resulting from combining the other two. In particular, there has been much research on extending the conventional object-oriented model to the fuzzy object-oriented model, e.g. [19], [23], [7], [12], [33] (cf. [17]), [36] (cf. [35]) and [14], but two main problems, namely, multiple inheritance and uncertain inheritance, still remain unresolved.

With multiple inheritance, an object or a class can inherit properties, i.e., attributes and methods, from two or more of its super-classes (cf. [6]). In the conventional object-oriented model, due to the procedural interpretation of methods, multiple inheritance raises the problem of deciding which method among those inherited of the same name is to be executed. Based on such a model, [23], [7], [36] and [14], did not discuss the problem, while [19] and [33] considered only multiple inheritance of attributes. In [12], the solution in the conventional object-oriented model was adopted, whereby the selected method was the one of the first super-class found in some ordering. As argued in [30], such an ordering-based selection was not sufficient.

Meanwhile, that selection problem disappears from the viewpoint of logic programming. Indeed, in the logic-based object-oriented model, a property can be represented by a clause, where an attribute is represented by a fact and a method by a rule. An object can be equated with a set of clauses representing all the properties that the object can have. As properties of the same name are just clauses whose heads are predicates of the same name, which are usual in logic programming, an object can inherit all properties of the classes to which it is a member. A message sent to an object can be interpreted as a query on its corresponding set of clauses, then the usual Prolog ([11]) non-deterministic and backtracking searching

¹ Contact author.

mechanism will find all possible answers to the query (cf. [30]). There can be conflict among the answers, and it is a role of soft computing in the partnership to resolve it.

Further, in the fuzzy object-oriented model, where an object may not certainly be a member of a class, there is another problem which is one of uncertain inheritance. It was called weighted inheritance in [33], but was not solved therein, nor addressed in the other works cited above on fuzzy extension of the conventional object-oriented model. This was also due to the procedural interpretation of methods, for which it was not clear what an execution of a method with some uncertain degree could mean. From the logic programming viewpoint, it simply means that the rule representing the method is weighted and applied with that uncertain degree, just as in a fuzzy logic program.

The work on Fril++ ([2], [3], [5]) is among a few attempts towards soft computing object-oriented logic programming. It extends Fril ([4]), which is a logic programming language that can handle both fuzzy and probabilistic uncertainties, with object-oriented features. However, the work still lacks a coherent framework for the development of Fril++ to be completed. Another related work is one on fuzzy order-sorted logic programming, which can handle uncertainty about types of objects ([9], [10]). In fact, (fuzzy) order-sorted logic programming and (fuzzy) object-oriented languages are closely related, where the former provides a logical basis for the later with classes being treated as types and inheritance as type unification. Fuzzy order-sorted logic programming, however, considers only fuzzy type labels but not class structures.

The following sections of this paper elaborate on the ideas introduced above for soft computing object-oriented logic programming. Section 2 presents a logic-based representation of classes. Section 3 discusses methods for evaluating uncertain membership of an object to a class. Section 4 addresses the uncertain and multiple inheritance problems and proposes solutions to them. Since in this paper our attention is focused on fundamental issues rather than a concrete language, we do not define a formal syntax yet but use a Prolog-like language (cf. [30]) to present the ideas. Section 5 outlines an application of the framework to object-oriented extension of fuzzy logic programming systems. Finally, Section 6 presents concluding remarks of the paper and suggestions of future research.

2. Class representation and hierarchy

In this section, a logic-based representation of class properties and their uncertain applicability to a class is presented. Then a relationship between classes in a hierarchy is discussed and established.

2.1. Representation of attributes and methods

A class is represented by a finite set of *properties*. A property is either an *attribute* or a *method*. Each attribute is associated with a *range* of values, defined on a domain,

that the attribute can take. If there is no uncertainty about the values that the attribute can take on the domain, this range can be defined as a subset of the domain. Generally, for the case with uncertainty, we assume it to be a fuzzy set interpreted as a possibility distribution on the domain. A crisp range can be considered as a special fuzzy set whose membership function has value 1 for the elements in the range and value 0 for the other elements in the domain of the fuzzy set. An attribute and its associated range are then represented by a fact, i.e., a Horn-like clause ([27]) whose body is empty.

For an example, let us consider the attribute *age* of the class YOUNG_PERSON. One may define the range of this attribute to be the subset [0, 30] of the domain [0, 120]. However, since *young* is intrinsically a vague concept ([22]), in most contexts there is not such a clear-cut boundary between *young* ages and *not young* ages. Rather, the range is a fuzzy set on [0, 120], which can be denoted by the linguistic label *young* and whose membership function can be defined by the diagram in Figure 2.1, for instance. The attribute *age* of the class YOUNG_PERSON with this range is then represented by the following fact: $\text{age}(\text{young})$.

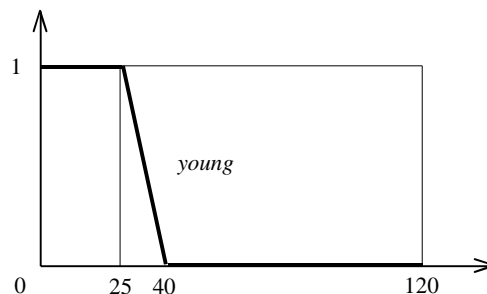


Figure 2.1. Fuzzy set *young*

We note that the range we define here for an attribute of a class corresponds to *necessary range* in [33], which also considered the *typical range* of typical values of an attribute. Meanwhile, in [14], only typical ranges were defined for attributes of a class. Although, these two kinds of ranges are conceptually different, there is no technical difficulty with having both of them for an attribute.

A method is represented by a rule, i.e., a Horn-like clause whose body is not empty, which can contains fuzzy set values. For example, the following rule represents a method of the class TOMATO saying “If a tomato is *red*, then it is *ripe*”:

$\text{ripeness}(\text{ripe}) :- \text{colour}(\text{red})$

where *ripe* and *red* are linguistic labels of fuzzy sets. We note that, in the object-oriented programming paradigm, the object that an attribute is referred to is context-dependent. That is, for example, the attributes *ripeness* and *colour* in the rule above are those of a tomato that the rule is applied to.

2.2. Uncertain applicability of properties

In the conventional object-oriented model, the properties that represent a class are necessary and sufficient to define the class. Arguing for flexible modelling, [36] introduced the notion of *fuzzy property* as an intermediate between the two extreme notions of required property and optional property. Each fuzzy property of a class was associated with possibility degrees of applicability of the property to the class.

Recently, [14] addressed the issue by contrasting the prototype concept model with the classical one. A severe defect of the classical concept model was noted by the fact that there was no commonly agreed set of defining (i.e., necessary and sufficient) properties for many natural, scientific, artificial and ontological concepts. Rather, each property of a concept was assumed to have a probability degree for it occurring in exemplars of the concept.

In the proposed framework, for its application to different existing fuzzy logic programming systems as it will be presented in Section 5, we provide two schemes for representation of uncertain applicability of properties to a class:

1. The *probability-based* scheme in which uncertain applicability of properties is represented by support pairs defining probability lower and upper bounds as in Fril ([14]).
2. The *possibility-based* scheme in which uncertain applicability of properties is represented by certainty (i.e., necessity) lower bounds as in possibilistic logic ([16]).

For example, the following attribute of the class BIRD:
fly (0.9 0.95)

where (0.9 0.95) is a support pair, expresses that the probability for the attribute *fly* being applicable to the class BIRD is in between 0.9 and 0.95 or, less formally, “90% to 95% of birds can fly”.

For another example, the class CAR may be modelled with the following attribute:

fuel_tank(*about 25 litres*) 0.9

where 0.9 is a certainty lower bound and *about 25 litres* is a linguistic label of a fuzzy set, to express “It is certain at least to degree 0.9 that a car has a fuel tank with the capacity of *about 25 litres*”.

As such, a support pair or a certainty lower bound associated with an attribute of a class reflects uncertain applicability of both the attribute and its range to objects of the class (cf. [14]). Uncertain applicability of methods will be exemplified in Section 3 and Section 4. For simplicity, in the examples in this paper, the support pair (1 1) and the certainty lower bound 1 are omitted.

2.3. Class hierarchy

In the conventional object-oriented model, a class hierarchy defines the subclass/super-class partial order (i.e., a reflexive, anti-symmetric and transitive relation) on classes. A class A is derived as a subclass of a class B , which is then called A 's super-class, either by narrowing the crisp ranges of B 's attributes or by adding new properties to B 's ones. Every member of A is also a

member of B or, in other words, the extension of A is included in that of B .

In the fuzzy case, due to the imprecision of attribute ranges or the uncertain applicability of class properties, the inclusion between classes naturally becomes graded. An inclusion degree of one class to another can be computed on the basis of the ranges of their common attributes. For example, in [33], four inclusion degrees were defined, depending on whether necessary ranges or typical ranges were used for each of the two classes.

As such, in general, if a class A has some inclusion degree to a class B , then B also has some inclusion degree to A . Thus, we argue that the notion of class “hierarchy” used in [19], [33] and [36], for instance, was not appropriate, because a set of classes with a graded inclusion or inheritance relation as defined therein formed a *network* rather than a *hierarchy*. This explains why the graded generalisation relation introduced in [23] was required to be anti-symmetric in the sense that, for every two concepts A and B , if $\mu(A, B) > 0$ and $\mu(B, A) > 0$ then $A = B$, where μ was the membership function of the relation.

However, in practice, one rarely has a hierarchy of classes with such a graded anti-symmetric generalisation relation, because the defined graded anti-symmetry implies that, for every pair of different classes A and B , if $\mu(A, B) > 0$ then $\mu(B, A) = 0$. Furthermore, naturally, a concept is usually classified into sub-concepts that are totally subsumed by it, though the sub-concepts can overlap each other. Therefore, in the proposed framework, for a class hierarchy, we assume that a class is totally subsumed by any of its super-classes or, in other words, a class totally subsumes any of its subclasses. This was also implicitly assumed in [5] and [14].

For an example adapted from [3], the class PERSON can have four subclasses BABY, CHILD, TEENAGER, ADULT. The ranges of the attribute *age* of these four subclasses are (fuzzy) subsets of the range of the attribute *age* of the class PERSON, which can be [0, 120] for instance. However, the (fuzzy) set intersection of these ranges may not be empty, i.e., there may be overlapping among the four subclasses .

3. Object representation and membership

This section presents a logic-based representation of objects, then two methods are discussed for evaluating uncertain membership of an object to a class. One method is based on class hierarchies and the other on class and object descriptions.

3.1. Object representation

As mentioned in Section 1, an object can be equated with a set of Horn-like clauses representing its properties. This is called the *clauses view* approach to object-oriented extension of logic programming ([13]). As in [30], we apply this approach in the proposed framework. The properties of an object comprises its private properties and

those inherited from its super-classes. Only the private properties, but not the inherited ones, need to be stored in the object’s knowledge base.

For an example, suppose that John is a student and the class STUDENT is a subclass of the class PERSON. This can be declared by using the notation in [30] as follows:

```
STUDENT <=PERSON
john <=STUDENT.
```

Then, besides the properties inherited from STUDENT and PERSON, a knowledge base for John may contain the following properties:

```
age(very young)
student_id(0123)
buy(X) :- like(X), X: price(not expensive) 0.7.
```

Here the rule says “It is certain at least to degree 0.7 that John buys a thing if he likes it and its price is *not expensive*”, where *not expensive* is a linguistic label of a fuzzy set and the attribute *price* is referred to the object represented by X.

In the fuzzy object-oriented model, the membership of an object to a class is not a matter of “to be or not to be” as in the conventional model, but rather a matter of degree. It seems that there is no universally applicable method for evaluating the membership degree of an object to a class on the basis of their fuzzy and uncertain descriptions. It may also require a learning process to obtain such a method in each particular domain.

Therefore, as in [3], for flexibility, we propose that each class has a user-defined method for computing the membership degree of an object to the class. Nevertheless, there are good default definitions for such an uncertain membership evaluation method, as presented below. Depending on whether the probability-based scheme or the possibility-based scheme of the proposed framework is applied, uncertain membership is measured by support pairs or certainty lower bounds, respectively.

3.2. Hierarchy-based membership evaluation

The hierarchy-based uncertain membership evaluation relies on the following assumptions:

1. If an object is a member of a class with some *positive characteristic* degree, then it is a member of any super-class of that class with the same degree.
2. If an object is a member of a class with some *negative characteristic* degree, then it is a member of any subclass of that class with the same degree.

These assumptions are a generalisation of ones stated in [10] and applied in [9] for fuzzy truth degrees, i.e., fuzzy sets on [0, 1], where the positive and negative characteristics were defined to be **true** and **false** characteristics, respectively. Other examples of positive characteristic degrees are probability or certainty lower bounds, while probability upper bounds are negative characteristic degrees.

As a consequence of the assumptions above, one has the following rules for uncertain membership evaluation:

1. For the probability-based scheme, if an object is a member of a class with a support pair ($l \ u$), then it is a member of any super-class of that class with the support pair ($l \ 1$), and a member of any subclass of that class with the support pair ($0 \ u$).
2. For the possibility-based scheme, if an object is a member of a class with some certainty lower bound, then it is a member of any super-class of that class with the same certainty lower bound.

These rules are in agreement with [33] and [5], for instance, which stated that the membership degree of an object to a class was at least equal to its membership degree to a subclass of that class. In [33], uncertain membership was measured by certainty degrees whereas, in [5], it was measured by probability degrees.

For examples, suppose the following subclass/super-class relations:

```
PENGUIN <=BIRD
BIRD <=WINGED_ANIMAL.
```

The fact “It is probable to a degree in between 0.8 and 0.9 that Billie is a bird” can be denoted as follows:

```
billie <=BIRD (0.8 0.9)
```

from which one can derives:

```
billie <=ANIMAL_WINGED (0.8 1) and
billie <=PENGUIN (0 0.9).
```

Meanwhile, the fact “It is certain at least to degree 0.95 that Penny is a penguin” can be denoted by:

```
penny <=PENGUIN 0.95
```

from which one can derives:

```
penny <=BIRD 0.95.
```

3.3. Description-based membership evaluation

The description-based uncertain membership evaluation relies on class attribute ranges, object attribute values and attribute applicability degrees. The intuition is that, if the class PERSON, for example, is classified into the subclasses BABY, CHILD, TEENAGER and ADULT by the attribute *age*, then it is rational to compute the membership degree of John to TEENAGER by comparing John’s age with TEENAGER’s range of the attribute *age*. This is actually a weighted fuzzy pattern matching problem.

For the possibility-based scheme, a method for computing the certainty lower bound for an object being a member of a class, without uncertain applicability of attributes, was thoroughly studied in [33]. Let R_1, R_2, \dots, R_n be the ranges of the attributes A_1, A_2, \dots, A_n of a class C , and V_1, V_2, \dots, V_n be the values of A_1, A_2, \dots, A_n of an object O . The certainty lower bound for O being a member of C was defined by:

$$N(C | O) = \min_{i=1,n} \{N(R_i | V_i)\} \quad (3.3.1)$$

where each $N(R_i | V_i)$ defined the relative necessity degree of the fuzzy set R_i given the fuzzy set V_i , which were assumed to be on the same domain.

In the case of uncertain applicability of A_1, A_2, \dots, A_n to C measured by certainty lower bounds c_1, c_2, \dots, c_n , respectively, a weighted version of (3.3.1) can be obtained on the basis of [18], which was also applied in [21]:

$$N(C | O) = \min_{i=1,n} \{ \max\{1 - c_i, N(R_i | V_i)\} \} \quad (3.3.2)$$

It satisfies the intuition that, if $c_i = 0$ then A_i does not effect $N(C | O)$, and if $c_i = 1$ then A_i contributes $N(R_i | V_i)$ to $N(C | O)$ as in (3.3.1).

For the probability-based scheme, let $(l_1 u_1), (l_2 u_2), \dots, (l_n u_n)$ be respectively the support pairs for applicability of A_1, A_2, \dots, A_n to C . We assume that each $(l_i u_i)$ gives A_i a weight w_i in evaluation of the membership of O to C , which is defined by:

$$w_i = (l_i + u_i) / \sum_{i=1,n} (l_i + u_i) \quad (3.3.3)$$

so that $\sum_{i=1,n} w_i = 1$. When $l_i = u_i$ for every i from 1 to n , (3.3.3) reduces to the definition of weights applied in [14], where uncertain applicability of properties was represented by single probability degrees.

Then, applying the evidential logic matching method in [4], the support pair for O being a member of C can be defined as follows:

$$(F(\sum_{i=1,n} w_i i)) \quad F(\sum_{i=1,n} w_i i)) \quad (3.3.4)$$

where $(i \quad i)$ is the support pair obtained by the *semantic unification* of R_i to V_i , based on *mass assignment theory* ([1]), and $F: [0, 1] \rightarrow [0, 1]$ is some function whose choice determines the nature of the combination of A_1, A_2, \dots, A_n . Choosing $F(x) = x$ for every $x \in [0, 1]$ corresponds to the way of combining properties in [14].

3.4. Combined membership evaluation

Like other fuzzy and uncertain data, membership degrees derived from different sources, such as the hierarchy-based membership evaluation and the description-based membership evaluation, can be combined to obtain less uncertain ones. Basic combination rules are as follows:

1. In the probability-based scheme, membership degrees represented by support pairs can be combined, as in Fril ([4]), by using either the intersection rule as the default rule, or Dempster rule when the prior membership degrees are derived from sources of independent and possibly conflicting viewpoints.
2. In the possibility-based scheme, if an object O is known to be a member of a class C with degree l_1 from one source and with degree u_2 from another source, where l_1 and u_2 are certainty lower bounds, then $\max\{l_1, u_2\}$ can be asserted as a membership degree of O to C .

For an example, let A be a subclass of B and B be a subclass of C . Suppose that an object O is known to be members of A and C with support pairs $(l_1 u_1)$ and $(l_2 u_2)$, respectively. By the hierarchy-based membership evaluation, $(l_1 1)$ and $(0 u_2)$ are support pairs for O being a member of B . Then, using the intersection rule, $(l_1 u_2)$ can be asserted as a membership degree of O to B , provided that $l_1 \leq u_2$. If $l_1 > u_2$, Dempster rule can be applied to resolve the conflict.

In the description-based membership evaluation method presented above, values of attributes of an object

are assumed to be readily known. However, in the knowledge base of an object, there are often more than one clause defining the value for an attribute of the object and, moreover, possibly with uncertain applicability degrees. For example, the knowledge base of a person John may contain the following facts about his age:

age(not young) (0.8 1)

age(not old) (0.7 0.9).

Depending on a particular fuzzy logic theorem prover which the proposed framework is applied to, there are two mechanisms for uncertain membership evaluation with multiple definitions of object attribute values. One is that the combined value of each attribute is deduced before the description-based membership evaluation method is applied. An alternative is that the method is applied to different values deduced from different proof paths for each attribute, then the resulting membership degrees are combined into a single membership degree.

4. Uncertain and multiple inheritance

In this section, the problems of uncertain and multiple inheritance in fuzzy object-oriented systems are discussed and logic-based solutions to them are proposed.

4.1. Uncertain inheritance

In the conventional object-oriented model, without exceptions, a class fully inherits all the properties of its super-classes and thus an object certainly has all properties of the classes to which it is a member. In the fuzzy object-oriented model, due to uncertain applicability of a property and uncertain membership of an object to a class, inheritance becomes uncertain. However, this problem has not been adequately solved in previous works.

In [19], [23] and [7], uncertainty in class hierarchies was studied but uncertain inheritance was not discussed. In [33], inclusion degrees between classes were defined, but inheritance was considered only in the case a class was certainly a subclass of another class. In [36], the membership degree of an object to a class was compared with the applicability degree of a property of the class, in order to decide whether the object could inherit the property. However, the membership degree was then not counted in the applicability degree of the inherited property to the object. Meanwhile, [14] computed membership degrees of objects to classes only for their concept recognition rather than for uncertain inheritance.

That common shortcoming was mainly due to the procedural interpretation of methods, as implicitly assumed in the works cited above, for which it was not clear what an execution of a method with some uncertain degree could mean. From the logic programming viewpoint, it simply means that the rule representing the method is weighted and applied with that uncertain degree, just as in a fuzzy logic program. This viewpoint was applied and exemplified in [2], [3] and [5] which, however, did not consider uncertain applicability of properties to a class.

In the proposed framework, we assume that a property declared in a class is (certainly) applicable to an object if and only if it is applicable to the class *and* the object is a member of the class. Consequently, the applicability degree of a property p of a class C to an object O can be defined as follows:

1. In the probability-based scheme, if $(l \ u)$ is the applicability degree of p to C and $(\)$ is the membership degree of O to C , then the applicability degree of p to O is $(\ l \ u)$, on the basis of probability theory.
2. In the possibility-based scheme, if c is the applicability degree of p to C and $\$ is the membership degree of O to C , then the applicability degree of p to O is $\min\{ \ , c\}$, on the basis of possibility theory.

As in [30], overriding inheritance and differential inheritance can also be applied, so that a class does not inherit some properties from its super-classes. For example, the class PENGUIN may have the following property:

fly (0 0)

to overrides the inherited property *fly* from its super-class BIRD. This subclass/super-class relation between PENGUIN and BIRD with overriding inheritance can be denoted by:

PENGUIN << BIRD.

An alternative is to declare PENGUIN as a subclass of BIRD with differential inheritance, so that PENGUIN inherits all BIRD's properties except for *fly*, as follows:

PENGUIN <= BIRD - [fly].

4.2. Multiple inheritance

For multiple inheritance, the procedural interpretation of methods also raises the problem of deciding which method among those inherited of the same name is to be selected for execution. In [23], [7], [36] and [14], multiple inheritance was not discussed while, in [19] and [33], only multiple inheritance of attributes was considered. Meanwhile, from the logic programming viewpoint, that selection problem disappears, as discussed and exemplified in [30] on logic-based object-oriented programming and in [2], [3] and [5] on its fuzzy extension.

Indeed, as presented in Section 3, in the logic-based object-oriented model, each object is viewed as a knowledge base of clauses representing all the properties that the object can have. Then inherited properties of the same name are included in the object's knowledge base just as clauses whose heads are predicates of the same name, which are usual in logic programming. A message sent to the object is viewed as a query on the object's knowledge base, which is answered by a Prolog style proof procedure. There can be different and, moreover, possibly conflicting answers to the query, and it is a role of soft computing to resolve the conflict and combine the answers. The solution depends on a particular fuzzy logic theorem prover being applied.

Let us take an example adapted from [2] that, there is a shape which resembles both a circle and a square, as illustrated in Figure 4.1, and one wants to evaluate its area approximately.

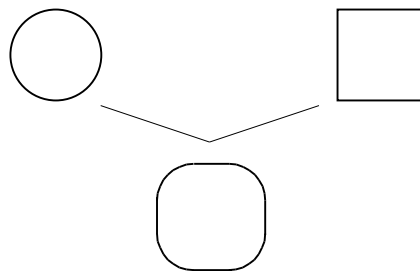


Figure 4.1. A shape resembling a circle and a square

The problem can be solved by considering the shape as members of both the class CIRCLE and the class SQUARE with some support pairs. The class CIRCLE contains the attribute *radius* and the method *area*, which computes the area of a circle and is represented by the following rule:

area(A) :- radius(R), A = 3.14 R R

where A and R respectively denote the area and the radius of a circle object that the method applies to. The class SQUARE contains the attribute *length* and the method *area*, which computes the area of a square and is represented by the following rule:

area(A) :- length(L), A = L L

where A and L respectively denote the area and the length of a square object that the method applies to. Here, A , R and L are fuzzy number variables, whose values are fuzzy sets on the set of all real numbers, and $\$ is a fuzzy multiplication operator (cf. [25]).

Suppose that the shape is given the support pair (0.7 0.8) for being a circle with the radius of *about 1.2* (cm), and the support pair (0.6 0.7) for being a square with the length of *about 2.2*, where *about 1.2* and *about 2.2* are linguistic labels of the fuzzy numbers [1.0: 0 1.2: 1 1.4: 0] and [1.8: 0 2.2: 1 2.6: 0], respectively, whose diagrams are given in Figure 4.2.

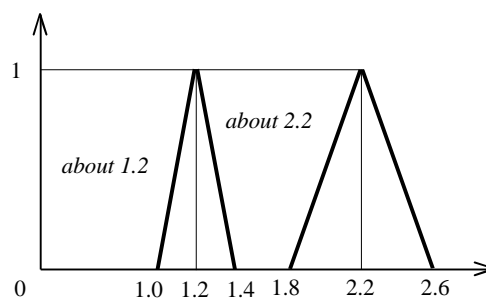


Figure 4.2. Fuzzy numbers *about 1.2* and *about 2.2*

The knowledge base of the shape then virtually contains the following clauses:

radius(*about 1.2*)

length(*about 2.2*)

area(A) :- radius(R), A = 3.14 R R (0.7 0.8)

area(A) :- length(L), A = L L (0.6 0.7)

where the two rules are inherited from the classes CIRCLE and SQUARE, and only the two facts are actually stored in the knowledge base. A message sent to this shape object to compute its area is interpreted as the query $area(A)$ on this knowledge base.

Implementing the knowledge base as a Fril program and running it in Fril with the query $area(A)$, one obtains as an intermediate result the following two separated answers corresponding to the two rules:

$$A = [3.14: 0 \quad 4.5216: 1 \quad 6.1544: 0] \quad (0.7 \quad 0.8)$$

$$A = [3.24: 0 \quad 4.84: 1 \quad 6.76: 0] \quad (0.6 \quad 0.7).$$

Fril then finds the expected fuzzy set of these two support pair-weighted fuzzy sets and defuzzifies it, producing the final result $A = 4.65336 \text{ (cm}^2\text{)}$ as an approximate area of the given shape.

5. Application

The proposed framework can be applied to extend fuzzy logic programming to fuzzy object-oriented logic programming. Fuzzy logic programming systems can be roughly classified into two groups with respect to whether they involve fuzzy sets in programs or not. Systems that do not involve fuzzy sets usually have formulas weighted by real numbers in the interval $[0, 1]$, interpreted as truth or uncertainty degrees, e.g. [31], [15] and [24]. Systems that involve fuzzy sets, which we call *fuzzy set logic programming*, include those of [34], [4], [20], [37] and [8]. Only fuzzy set logic programming systems can be used with the proposed framework to represent and reasoning with fuzzy set values in class and object properties.

In systems like [20] and [8], program clauses are Horn-like clauses weighted by certainty lower bounds in the following form:

$$H \quad B_1, B_2, \dots, B_n \quad [\quad]$$

where H, B_1, B_2, \dots, B_n represent fuzzy set values in the head and the body of the clause, and $[\quad]$ represents the certainty lower bound for the clause being **true**. Inference is then performed through certainty degree propagation, qualification and modification. The developed possibility-based scheme is intended to be used with such fuzzy set logic programming systems.

Meanwhile, in the system of [4], which is named Fril, a program comprises Horn-like clauses weighted by support pairs, with the following as an example of a rule:

$$\begin{aligned} & ((\text{illness of X is flu}) \\ & \quad ((\text{temp of X is high}) (\text{strength of X is weak}) \\ & \quad \quad (\text{throat of X is sore})) \\ &) \\ &): (0.9 \quad 1) \end{aligned}$$

where *high*, *weak* and *sore* are linguistic labels which can be defined by fuzzy sets. The rule expresses ‘‘A person has flu, if his/her temperature is *high*, strength is *weak*, and throat is *sore*’’ with the support pair $(0.9 \quad 1)$.

A support pair attached to a clause gives the lower bound and the upper bound of the probability for the clause being **true**. For a rule, it is the conditional probability of the head given the body of the rule. Inference is performed through semantic unification of

vague data to derive a support pair for a conclusion, based on mass assignment theory, which integrates both probability theory and possibility theory and provides methods to deal with combined uncertainty degrees of different characteristics. The developed probability-based scheme is intended for extending Fril with object-oriented features, some basic ideas of which have been discussed in [2], [3] and [5].

The main methodology of the application of the proposed framework for object-oriented extension of fuzzy set logic programming systems is that each object is viewed as a fuzzy logic program. The program comprises clauses representing properties of the object, which are weighted by certainty lower bounds or support pairs representing uncertain applicability of the properties. A message sent to an object is viewed as a query on the program representing the object and is answered by the theorem proving mechanism of the underlying fuzzy logic programming system.

6. Conclusion

We have presented a logic-based fuzzy object-oriented framework that combines logic programming, object-oriented programming and soft computing. In the framework, class and object properties are represented by Horn-like clauses. The clauses can contain fuzzy set values to represent vague data in the properties and can be weighted to represent uncertain applicability of the properties. We have provided two schemes, namely, probability-based and possibility-based, for representing uncertain applicability of properties.

In both of the schemes, we have proposed the hierarchy-based and the description-based methods as good default methods for evaluating uncertain membership of an object to a class. Uncertain inheritance problem has then been discussed and its solution has been proposed, whereby the applicability degree of a property of a class to an inheriting object is defined by the applicability degree of the property and the membership degree of the object to the class. It has shown that the problems of uncertain and multiple inheritance seen from the imperative programming viewpoint disappears from the logic programming viewpoint.

We have outlined how the framework can be applied to object-oriented extension of existing fuzzy set logic programming systems with probability or certainty weighted program clauses. In particular, we are applying it to extend Fril to Fril++ as a language tool for developing soft computing object-oriented knowledge bases. A Fril-based formal syntax will have to be defined for representation of object-oriented features. A mechanism for managing local knowledge bases of objects will have to be designed. With uncertain inheritance, in principle, an object can inherit properties from any class, not as in the conventional model where it inherits properties only from its super-classes. As such, for efficiency, a searching strategy will have to be studied for pruning off inheritance paths leading to properties with too low applicability

degrees to be inherited. These are among the topics that are currently being investigated.

References

- [1] Baldwin, J.F. 1992. Fuzzy and probabilistic uncertainties. In Shapiro, S.C. (ed.), *Encyclopedia of Artificial Intelligence* (2nd edition), John Wiley & Sons, pp. 528-537.
- [2] Baldwin, J.F. and Martin, T.P. 1995. Refining knowledge from uncertain relations - a fuzzy data browser based on fuzzy object-oriented programming in Fril. In Proceedings of the 4th IEEE International Conference on Fuzzy Systems, pp. 27-34.
- [3] Baldwin, J.F. and Martin, T.P. 1996. Fuzzy classes in object-oriented logic programming. In Proceedings of the 5th IEEE International Conference on Fuzzy Systems, pp. 1358-1364.
- [4] Baldwin, J.F., Martin, T.P. and Pilsworth, B.W. 1995. *Fril - Fuzzy and Evidential Reasoning in Artificial Intelligence*. Research Studies Press.
- [5] Baldwin, J.F., Martin, T.P. and Vargas-Vera, M. 1998. Fril++: object-based extensions to Fril. In Martin, T.P. and Fontana, F.A. (eds), *Logic Programming and Soft Computing*, Research Studies Press, pp. 223-238.
- [6] Booch, G. 1994. *Object-Oriented Analysis and Design with Applications* (2nd edition). Addison-Wesley.
- [7] Bordogna, G., Lucarella, D. and Pasi, G. 1994. A fuzzy object oriented data model. In Proceedings of the 3rd IEEE International Conference on Fuzzy Systems, pp. 313-318.
- [8] Cao, T.H. 1998. Annotated fuzzy logic programs. *International Journal for Fuzzy Sets and Systems*. Accepted for publication. To appear.
- [9] Cao, T.H. and Creasy, P.N. 1998. Fuzzy order-sorted logic programming in conceptual graphs with a sound and complete proof procedure. In Mugnier, M.L. and Chein, M. (eds), *Conceptual Structures: Theory, Tools and Applications*, LNAI 1453, Springer-Verlag, pp. 270-284.
- [10] Cao, T.H., Creasy, P.N. and Wuwongse, V. 1997. Fuzzy types and their lattices. In Proceedings of the 6th IEEE International Conference on Fuzzy Systems, pp. 805-812.
- [11] Clocksin, W.F. and Mellish, C.S. 1987. *Programming in Prolog* (3rd, revised and extended edition). Springer-Verlag.
- [12] Cross, V. 1996. Towards a unifying framework for a fuzzy object model. In Proceedings of the 5th IEEE International Conference on Fuzzy Systems, pp. 85-92.
- [13] Davison, A. 1993. A survey of LP-based object oriented languages. In Wegner, P., Yonezawa, A. and Agha, G. (eds), *Research Directions in Concurrent Object Oriented Programming*, MIT Press, pp. 42-106.
- [14] Dubitzky, W., Büchner, A.G., Hughes, J.G. and Bell, D.A. 1999. Towards concept-oriented databases. *Data & Knowledge Engineering*, 30, 23-55.
- [15] Dubois, D., Lang, J. and Prade, H. 1991. Towards possibilistic logic programming. In Proceedings of the 8th International Conference on Logic Programming, MIT Press, pp. 581-595.
- [16] Dubois, D., Lang, J. and Prade, H. 1994. Possibilistic logic. In Gabbay, Dov M. et al. (eds), *Handbook of Logic in Artificial Intelligence and Logic Programming*, Vol. 3, Oxford University Press, pp. 439-514.
- [17] Dubois, D., Prade, H. and Rossazza, J-P. 1991. Vagueness, typicality and uncertainty in class hierarchies. *International Journal of Intelligent Systems*, 6, 167-183.
- [18] Dubois, D., Prade, H. and Testemale, C. 1988. Weighted fuzzy pattern matching. *International Journal of Fuzzy Sets and Systems*, 28, 313-331.
- [19] George, R., Buckles, B.P. and Petry, F.E. 1993. Modelling class hierarchies in the fuzzy object-oriented data model. *International Journal of Fuzzy Sets and Systems*, 60, 259-272.
- [20] Godo, L. and Vila, L. 1995. Possibilistic temporal reasoning based on fuzzy temporal constraints. In Proceedings of the 14th International Joint Conference on Artificial Intelligence, pp. 1916-1922.
- [21] Granger, C. 1988. An application of possibility theory to object recognition. *International Journal of Fuzzy Sets and Systems*, 28, 351-362.
- [22] Honderich, T. (ed.) 1995. *The Oxford Companion to Philosophy*. Oxford University Press.
- [23] Itzkovich, I. and Hawkes, L.W. 1994. Fuzzy extension of inheritance hierarchies. *International Journal of Intelligent Systems*, 62, 143-153.
- [24] Klawonn, F. 1995. Prolog extensions to many-valued logics. In Höhle, U. and Klement, E.P. (eds), *Non-Classical Logics and their Applications to Fuzzy Subsets*, Kluwer Academic Publishers, pp. 271-289.
- [25] Klir, G.J. and Yuan, B. 1995. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Prentice Hall.
- [26] Liu, M. and Guo, M. 1998. ROL2: a real deductive object-oriented database language. In Proceedings of the 17th International Conference on Conceptual Modeling, LNCS 1507, Springer-Verlag, pp. 302-315.
- [27] Lloyd, J.W. 1987. *Foundations of Logic Programming* (2nd, extended edition). Springer-Verlag.
- [28] Lukasiewicz, T. 1998. Probabilistic logic programming. In Proceedings of the 13th Biennial European Conference on Artificial Intelligence, pp. 388-392.
- [29] Martin, T.P. and Fontana, F.A. (eds) 1998. *Logic Programming and Soft Computing*. Research Studies Press.
- [30] McCabe, F.G. 1992. *Logic and Objects*. Prentice Hall.
- [31] Mukaidono, M., Shen, Z. and Ding, L. 1989. Fundamentals of fuzzy Prolog. *International Journal of Approximate Reasoning*, 3, 179-194.
- [32] Ng, R. and Subrahmanian, V.S. 1992. Probabilistic logic programming. *Information and Computation*, 2, 150-201.
- [33] Rossazza, J-P., Dubois, D. and Prade, H. 1997. A hierarchical model of fuzzy classes. In De Caluwe, R. (ed.), *Fuzzy and Uncertain Object-Oriented Databases: Concepts and Models*, World Scientific, pp. 21-61.
- [34] Umamo, M. 1987. Fuzzy set Prolog. In Preprints of the 2nd International Fuzzy Systems Association Congress, pp. 750-753.
- [35] Van Gyseghem, N., De Caluwe, R. and Vandenberghe, R. 1993. UFO: uncertain and fuzziness in an object-oriented model. In Proceedings of the 2nd IEEE International Conference on Fuzzy Systems, pp. 773-778.
- [36] Van Gyseghem, N. and De Caluwe, R. 1997. The UFO database model: dealing with imperfect information. In De Caluwe, R. (ed.), *Fuzzy and Uncertain Object-Oriented Databases: Concepts and Models*, World Scientific, pp. 123-185.
- [37] Virtanen, H.E. 1996. Lukasiewicz logic programming based on fuzzy equality. In Proceedings of the 4th European Congress on Intelligent Techniques and Soft Computing, pp. 646-650.
- [38] Zadeh, L.A. 1997. The roles of fuzzy logic and soft computing in the conception, design and development of intelligent systems. In Nwana, H.S. and Azarmi, N. (eds), *Software Agents and Soft Computing*, LNAI 1198, Springer-Verlag, pp. 183-190.