Fuzzy Logic for the Retrieval Step of a Case-Based Reasoner

Michel Jaczynski and Brigitte Trousse

INRIA Sophia-Antipolis
2004, route des Lucioles, BP 93
06902 Sophia-Antipolis Cedex, France
{mjaczyn,trousse}@sophia.inria.fr

Abstract. This paper presents an original use of fuzzy logic for the design of Case-Based Reasoning (CBR) assistants, mainly for the case retrieval step. Our motivation comes from the importance of allowing for the expression of flexible data (incomplete and imprecise data, preferences) in problem and case descriptions and to manage them in case retrieval. We present our retrieval system CAREFUL based on possibility theory that takes into account this flexibility. For an efficient retrieval, we have adapted a hierarchical fuzzy classification algorithm for case filtering. This step can also be used to help the user to describe the problem, and enable him to specify its own similarity for each problem attribute. We have also adapted fuzzy pattern matching techniques in order to execute the selection step. Before comparing our approach to some related CBR works and concluding, we illustrate our system in a simple example.

1 Introduction

This paper shows the utility of fuzzy logic in a Case-Based Reasoning (CBR) context, focusing on the two first steps of a case-based reasoner: the construction of the problem specification and the case retrieval. First we present a short analysis of the flexible aspects of complex problem solving activities at the knowledge level (introduced by A. Newell in [19]). Based on our understanding of design activities in particular, we show the importance of allowing for incomplete and imprecise data, and preferences in the descriptions of cases and problems: our goal is to improve the cognitive validity of CBR assistants. Secondly we present – at the symbol level – our CBR retrieval system shell named CAREFUL (CAse REtrieval based on FUzzy Logic) that shows how fuzzy logic can be used to represent and process flexible data. Our system is presented according to two points: the problem and the case representation, and the retrieval process. A more detailed description of the system can be found in [11]. Then we show how our system works in a simple example. Finally, after comparing our approach to some related CBR works, we propose some perspectives that could enhance the possibilities of our system.

2 Analysis of Flexible Data in Problem Solving

Our assumption that flexible data must be taken into account is mainly derived from different concrete experiences in developing design support AI tools in Spatial Engineering [22, 23] as well as works in Cognitive Psychology [24, 25]. We first define the flexible aspects of problem solving, and then we identify the implications for a case-based reasoner.

2.1 Four Main Origins of Flexibility

An analysis of flexibility in design activities and more generally in problem solving activities shows four main origins: a/ the existence of explicit preferences between characteristic values and/or characteristics, b/ the incompleteness, the imprecision and the uncertainty of data, c/ the evolution of data during the problem solving and d/ the hierarchical aspect of the design process. For more details, see [12].

In our work we focus on the two first origins of this flexibility, which seem to us to be the most important:

- The preferences are very useful to describe design problems: for example, *"I prefer a white car rather than a red one, but a gray one is also good"* or *"I prefer to privilege the price attribute rather than the color one"*.
- The problem description may be imprecise (for example, *"the length of X is around 5 cm"*). The problem description can also be incomplete in regard to *a priori* defined types of problems because some data are not provided or are unknown.

Flexibility is very important for complex problem solving activities: for an example in planning, see the CARS system [3] and for an identification problem, see the FLORAN system [21]. It is also clear that in some domains such as physics, sociology, and economy [18], we need to consider flexibility aspects if we want to tackle real problems.

2.2 Implications for a Case-Based Reasoner

Focusing on the two first steps of a case-based reasoner, we analyze here the implications of taking flexibility into account in three features of a case-based reasoner: the target description and the user request, the case description and the retrieval process.

- The target description and the user request are mainly based on the problem description which is – as we have shown – flexible by nature in many different domains. In our work we wish to support novices during the target description in a domain where some problem types might be identified (for example in routine design). This support may reduce the incompleteness of the description.

- The case usually includes a problem description, which is – as we have already shown – flexible. But the other components of a case – such as the solution or the outcome – may be flexible too because of an imprecise or incomplete description in regard to a solution model. A case could also be imprecise because it represents an abstraction of several precise experiences.
- Usually, a case-based reasoner must find *a case similar to the target or the problem description*. Indeed most CBR systems are based on similarity relations between the target and the cases. These relations are vague by nature. The aim is not to find a case that satisfies crisp goals but to find a case close to the current context that best satisfies these goals. Thus, even if cases and target are precise and certain, the retrieval process involves flexibility. In any case, the retrieval process must take into account and propagate the flexibility expressed in the target and the cases.

3 Problem and Case Representation in CAREFUL

We will first show how we represent flexible data. Then we will present our object-oriented formalism, which is used to store *a priori* knowledge about the domain.

3.1 Flexible Data Representation based on Fuzzy Sets

Problems and cases could be represented by a list of attribute-value pairs where a value is a fuzzy set. We have chosen to use fuzzy sets in order to represent the exactness, the vagueness, the unknown and the preferences in an homogeneous way (see Fig. 1).

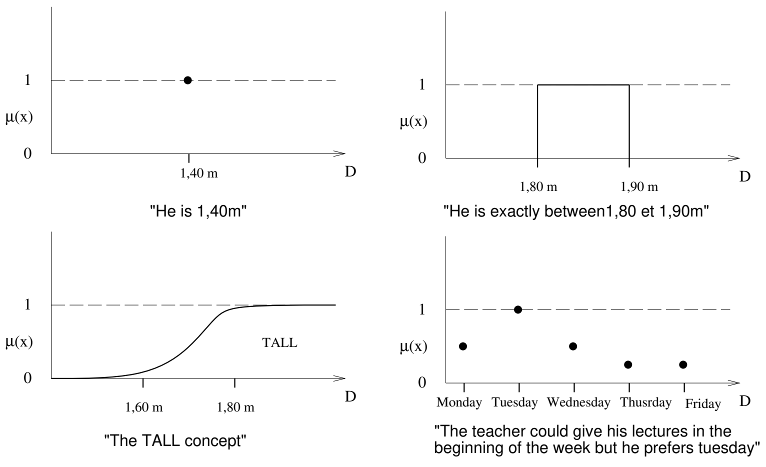

Fig. 1. Different interpretations of fuzzy sets.

Fuzzy sets have been introduced by Zadeh [26] in order to represent classes or sets of objects non-precisely defined where classical set theory cannot be used. A fuzzy set A in domain D is defined by a characteristic (membership) function μ_A that takes its values inside the $[0, 1]$ interval. This function represents the degree of membership of an element x of D into A. A characteristic function equal to 1 represents an unknown value.

In a case description, fuzzy values are used to represent vagueness but in the problem description, each value must define a membership constraint on the attribute domain. This constraint can be built directly by the user in order to represent preferences or it can be built from an observed exact or fuzzy value by using fuzzy relations. In other terms, the user is able to specify its own notion of similarity for each attribute. This is one of the original features of our approach.

3.2 Object-Oriented Knowledge Representation

In order to represent a part of the *a priori* knowledge about the domain needed to help the user in the problem description and also to guide and improve the retrieval process, we propose to use an object-oriented representation based on a simple inheritance hierarchy of fuzzy classes. A class is used to define a set of attributes. An attribute is described by a set of facets. The attributes and their facets are inherited in more specific classes. Our system can manage different hierarchies in order to represent different viewpoints of the same concept or different concepts but, for the moment, we don't manage composite description. Each hierarchy has to be built with regard to a specific viewpoint and to a specific outcome to maintain the consistency of the description. With this representation, a case or a problem is defined by a set of instances.

A class defines a fuzzy context because each attribute of that class has a `domain` facet and a `restriction` facet. The domain can be a float or integer interval, an ordered symbol list, a symbol set, or a symbol hierarchy. The restriction is represented by a fuzzy set and defines the values compatible with that class for the attribute in its domain. Restrictions can be refined in more specific classes in order to define a more precise context. Moreover, the existence and the relevance of an attribute may depend on the context, so this object-oriented representation allows one to add an attribute in a sub-class only when the context needs it. This means that the representation of a case or a problem depends on the context.

For each attribute, default values can be provided thanks to the `default` facet and the `symbolic-values` facet. The default value can be an element of the attribute domain or a method which is able to compute the default value depending on other attribute values. Symbolic values are defined by a set of couples (symbol, fuzzy set on the attribute domain). In our system, symbolic values (see Fig. 2) are used to define an appropriate vocabulary in the context.

An index is a particular attribute that is used to guide the retrieval process during the filtering step (see section 4.1) and during the selection step (see section 4.2). The set of indexes may differ from the filtering step to the selection

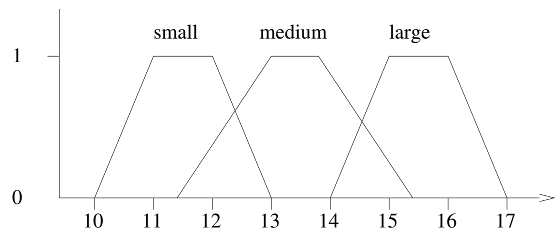

Fig. 2. Example of symbolic values in a real domain.

step. Indexes of the filtering step are always determined statically by an expert, and we use the two following facets: the `filtering-index` facet indicates if the attribute is an index, and the `filtering-importance` facet gives its importance. An index importance is defined by a symbol such as *very-important*, *important*, *less-important*, and so on. Each of these symbols is associated to a numerical value inside $[0, 1]$. But for the selection step, indexes are dynamically chosen, and we have defined two other facets (`default-selection-index` and `default-selection-importance`) to provide defaults or methods to compute these defaults.

Other facets have been defined for the user interface, such as the precedence of an attribute, the question text, the comments and a flag to indicate if the value of a slot has to be asked or computed[1].

An example of a hierarchical memory organization using our object-oriented and fuzzy-set based representation is described in section 5.1.

4 Case Retrieval in CAREFUL

We have divided the retrieval process into two steps:

- **the problem specification and case filtering step** which guides the operator in specifying the problem and isolates the potentially interesting cases,
- and **the selection step** which chooses precisely the nearest cases to be used.

Therefore, the retrieval is more efficient because the set of cases to review in the selection step is a subset of the entire case base.

4.1 Aid to Problem Specification and Case Filtering

The problem specification and case filtering step is based on an existing hierarchical fuzzy classification algorithm which we have adapted to a CBR context. We will first introduce the chosen hierarchical fuzzy classification and then we will show how we have adapted it for the case filtering.

[1] Computed values have not been implemented yet.

Use of a Hierarchical Fuzzy Classification. The hierarchical fuzzy classification is based on a hierarchy of fuzzy classes that defines attributes and their compatible values. The goal is to find the most specific classes that are compatible with a fuzzy object description. A concrete implementation of this kind of classification with some restrictions can be found in the CLASSIC expert system shell in classification [9]. This algorithm uses the results of the possibility theory introduced by Zadeh [27], based then on the computation of two numbers: the possibility and the necessity. These two numbers are used to evaluate the membership of a fuzzy value B into a fuzzy pattern A defined in a class for a particular attribute. Thus we have the following definitions, A and B being two fuzzy sets:

– the possibility:

$$\begin{aligned}\prod(A|B) &= \max_{x \in D}(\min(\mu_A(x), \mu_B(x))) \\ &= \text{degree of intersection between } A \text{ and } B\end{aligned}$$

– the necessity:

$$\begin{aligned}N(A|B) &= \min_{x \in D}(\max(\mu_A(x), 1 - \mu_B(x))) \\ &= \text{degree of inclusion of } B \text{ into } A\end{aligned}$$

We can now aggregate these numbers with the importance weight of each attribute as in [7], in order to obtain a global possibility and necessity for the class. In a fuzzy classification, the algorithm starts at the root of the hierarchy with an object description as input: it compares the class definition and the object description, it computes possibilities and necessities for each attribute, it combines them, and then, if these results are under the predefined thresholds, the class is rejected, otherwise the object is compatible with that class and the algorithm goes on to its subclasses (if there are any). We can also reject a class as soon as a possibility or a necessity of one attribute is under a threshold.

We propose to use a hierarchical fuzzy classification for the three following main concerns of a case-based reasoner. First, this classification algorithm is well suited to filter the cases. Indeed, if we classify a given problem into one or more problem classes, we will be able to obtain old cases which have been classified in these classes, and thus retrieve potentially interesting cases. Secondly, the problem can be classified in some classes even if the matching is not complete or exact because this algorithm takes into account fuzzy values of the problem description and propagates them. Finally, if we use an interactive version of a hierarchical classification algorithm, an operator can be guided dynamically by the questions asked by the computer about some attribute values. This is very interesting when problems are very complex and when the operator is not an expert in the domain.

Adapted Classification Algorithm for the Case Filtering. The originality of our work consists in the integration and adaptation of classification techniques into the retrieval step of a case-based reasoner. In our system we classify the current problem description in one hierarchy defined for the problem and case representation. We have mainly adapted the classification algorithm of CLASSIC on several points: the addition of links between old cases and their compatible classes in order to retrieve them, the introduction of the notions of index and index importance, and the direct specification of fuzzy sets.

Our algorithm starts at the root of the chosen hierarchy according to an outcome and a viewpoint. The system rejects or accepts the classes of the hierarchy with a depth first strategy in order to make a list of all compatible classes. Then we select the most specific classes from this list and retrieve their cases. The result of our case filtering algorithm is a list of couples (problem description, set of cases) because a problem could have been classified in more than one class.

The previous cases have been classified with this classification algorithm using the same set of static indexes with the same importances. The use of static indexes ensures that a previous case and a current problem which are identical will be classified in the same set of classes. A case could have been classified in several classes, and CAREFUL has linked the previous case description to each of these classes.

In order to accept or reject a class, CAREFUL executes the following steps:

1. The list L is made of redefined attributes and new attributes of the current class.
2. For each attribute element of L:
   (a) Get the value of the attribute.
   (b) If the attribute is an index, the possibility and the necessity are computed. If these numbers are under predefined thresholds, the class is REJECTED.
3. The class is ACCEPTED, and the global possibility and necessity are computed using local and inherited possibilities and necessities. These global degrees can be used to rank classes.
4. The indexes and their importance for the selection step are dynamically chosen among the attributes defined in the current class. These indexes can be totally different from the indexes used in the filtering step.

This algorithm can be used at three levels of interaction: automatic, manual and semi-automatic. At the automatic level, attribute values and indexes for the selection step are read in a declarative description made before the filtering. At the manual level, the system asks the user to provide attribute values. He can answer with a default value, a symbolic value or directly with a fuzzy set. If the class is accepted, the user is allowed to change the list of the proposed indexes for the selection step. At the semi-automatic level, the system first reads the information in the declarative problem description. If a value is unknown, the user is able to provide it. Thanks to these levels, CAREFUL is adaptable to different users with different levels of expertise.

4.2 Selection of Nearest Cases

After the filtering step, we obtain for each selected class a set of potentially interesting cases related to a problem description with a set of indexes. Then we must compare each case with the problem description in order to select the nearest cases. Each value of a problem attribute represents a membership constraint (a pattern), then we have to retrieve cases that best satisfy these constraints. In that sense, this problem is very close to fuzzy queries on fuzzy databases. Therefore, we use weighted fuzzy pattern-matching techniques described in [7].

In CAREFUL we do this selection in the context of each class identified by the filtering step. Thus, the system compares one problem description instance to each case instance of the same class and computes the global possibility and necessity using dynamic indexes defined during the filtering step. If the possibility and the necessity are under the predefined thresholds, the corresponding case is ignored. Of course, we make the assumption of a good coverage of the domain in order to select at least one case. If more than one case is selected, we are faced to the problem of ranking cases according to their global possibility and necessity. For this purpose CAREFUL sorts cases by their possibility and then by their necessity (other ranking methods could be used). Finally, the system is able to select the nearest cases for each class. It is also possible to rank all the cases in order to select the best case.

5 Illustration of CAREFUL in an Example

We will present here a simple example of an application of CAREFUL in order to give to the reader a better understanding of the originality of our system. We will first describe the hierarchical memory organization, then we will show an example of our retrieval process.

5.1 Hierarchical Memory Organization

We take the example of designing a car, and we present the car hierarchy according to the performance viewpoint. The class `Car-Performance` is the root of this hierarchy (see Fig. 3). The outcome attribute could represent the design plan or the design solution. We assume that performance description is relevant for this outcome and we also assume that the consumption is only relevant for this outcome when the car has an average performance. This example uses a case base of 30 cases. Each case is a fuzzy description of several versions of the same car that have nearly the same performance. These cases have been classified and linked to their compatible classes.

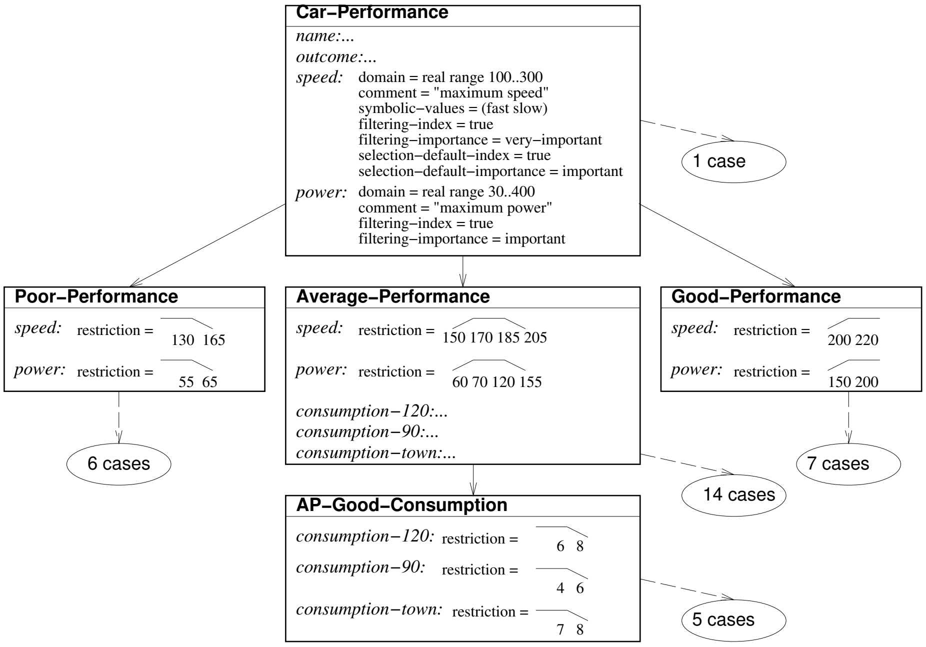

Fig. 3. A simple hierarchy definition.

5.2 Problem Specification and Case Retrieval

Now we show an example of the retrieval step at the manual interaction level starting at the class `Car-Performance`. The acceptance thresholds have been set to 0.3 for the possibility and 0. for the necessity. This means that a possibility under 0.3 will cause a class or a case to be rejected, and the necessity will not be used to reject or accept a class but just to rank cases. For each attribute CAREFUL asks the value to the user. The user can choose one of the symbolic values or the default value, or he can specify its own similarity with a fuzzy set. CAREFUL provides a lot of ways to define a fuzzy set using different keywords with the function `make-fuzzy-set`. The following dialogue is generated for the class `Car-Performance`:

```
What is the maximum speed?
domain: real range 100..300
default: unknown
symbolic values: (fast slow)
>>?(make-fuzzy-set min-tolerance: 175. min: 180. )
```

```
Maximum power of the car?
domain : real range 30..400
```
>>?*(make-fuzzy-set min-tolerance: 80. min: 90. max: 170. max-tolerance: 175)*

When all the values have been provided, and if they are compatible with the current class, the user is able to modify the table of indexes for the current class that will be used in the selection step:

```
Selection of indexes for the class Car-Performance:
----------------------------------------------------
name        index     importance     comment
----------------------------------------------------
power         t       important      Maximum power
speed         t       important      Maximum speed
>>?
```

At this time the system must check the sons of the current class. In our example, the class `Poor-Performance` is rejected and the class `Good-Performance` is accepted. The user provides the different attribute values for the class `Average-Performance` which is accepted but the class `AP-Good-Consumption` is rejected. Finally, CAREFUL retrieves 7 cases from the class `Good-Performance` and 14 from the class `Average-Performance`. The selection step compares these cases to the problem description in the context of each class and selects 3 cases from the class `Good-Performance` and 2 from the class `Average-Performance`.

5.3 Implementation

CAREFUL has been implemented using a powerful object-oriented environment named ICOS (meta-classes, classes, generic functions, meta-object protocol) under the Ilog Talk lisp dialect [10]. All the retrieval process is represented by a protocol of generic functions and controlled by system classes and objects. Thus, modifications can be easily supported by the specialization of the protocols and the system classes.

6 Related Works

In this section, we compare our approach with classical non-fuzzy indexing methods and also with three systems called ARC [20], FLORAN [21] and CARS [3]. The two main aspects used for the comparison are: problem and case representation, aid to the problem description, and case retrieval.

6.1 Non-Fuzzy Indexing Methods

Our retrieval techniques differ – of course – from all classical ones because we take flexible data into account. But it is interesting to compare our approach to non-fuzzy retrieval methods.

In comparison to MOP-based retrieval [13], we have abstraction and specialization links, and in a reduced manner index links thanks to restriction constraints. However, we don't have scene links because we don't manage composite cases for the moment.

We can also compare our techniques to the prototypical indexing method of the CBR shell ReMind [5]. Indeed, ReMind's prototype hierarchy and our class hierarchy define a context thanks to some constraints and they are linked to previous cases. However, the formalism in ReMind is totally different: cases and problems are not instances of prototypes and the set of attributes is always the same, even if some attributes are not relevant to a particular context.

An aid to problem description can be found during an inductive retrieval in ReMind or in Kate [15], but the attributes proposed to the operator are selected by means of an information gain criterion. In our system the expert has to define these attributes, and the operator is allowed to choose which attributes will be the indexes for the selection step.

6.2 The ARC System

In the ARC system [20], memory organization is based on a hierarchy of categories and cases. A category is formed by a fuzzy prototypical description of features common to most cases belonging to the category. Each category is linked to its sub-categories, a set of differentially-indexed cases, and near-misses cases. This organization is dynamic and categories can be modified or created after each reasoning. In CAREFUL the hierarchy is static because we want to represent static *a priori* knowledge.

The retrieval process first selects the most promising categories by a pattern-matching algorithm based on common features between the problem and the categories. Then cases are selected using differences between the categories and themselves. Near-misses cases are also used to retrieve past failures.

The main difference between ARC and CAREFUL is the management of flexibility. In ARC the fuzzy prototypical description and each matching degree computed during the retrieval, are based on a linguistically expressed uncertainty. In CAREFUL we use possibility theory and matching degrees are estimated in a more meaningful way by the possibility-necessity pair which represents an optimistic and a pessimistic estimation. Above all, fuzzy sets used in our system can represent imprecision and preferences.

6.3 The FLORAN System

In the FLORAN system, described in [21], cases and problems are also represented in an object oriented environment. Therefore cases and problems are instances of classes. Classes define slots that have a fuzzy value. FLORAN is able to manage a composite case and problem representation. Classes are linked to a set of dependency contexts. A dependency context is an object that represents for a given class of description, a specific goal, and a context made of fuzzy restrictions on each attribute, the list of relevant attributes with their importance and their tolerance. The main difference with our system is that the contexts are not organized in a hierarchy.

The retrieval process in FLORAN is also divided into a filtering step and a selection step, but the techniques employed differ from CAREFUL mainly in the filtering step. First, there is no aid to the problem description, the user must choose a class and a goal for the problem description and provide all fuzzy or exact data. Secondly, the system looks for the most compatible dependency context and get old cases associated to it. This step is a linear fuzzy classification of the problem into one or several contexts, whereas, in our system, a hierarchical fuzzy classification is done. Above all, some attributes may be provided but not used at all, because they never appear in a context, but in our system, attributes that are given by the operator are always necessary.

In the selection step, the current problem is compared with each filtered case using a fuzzy pattern matching algorithm where the problem is transformed into a pattern thanks to fuzzy tolerant relations as defined in [4]. The fuzzy relation used in FLORAN is:

$$\forall (x, y) \in D^2,\ \mu_{R_\lambda}(x, y) = \max(0, 1 - \frac{d(x, y)}{\lambda})$$

where λ is the tolerance defined in the selected context, and d is a distance on the domain D. Fuzzy tolerant relations are used to enlarge fuzzy description of the problem in order to build the pattern: for instance, an exact value is transformed into a fuzzy set of nearest values. The fuzzy pattern is defined by a membership function:

$$\forall x \in D,\ \mu_{A \circ R_\lambda}(x) = \sup_{y \in D} \min(\mu_A(y), \mu_{R_\lambda}(x, y))$$

where A is the original fuzzy (or exact) value. In fact, this means that the user cannot directly express membership constraints and value preferences as in our system. Another difference resides in the selection of indexes and the assignment of importance weights: in FLORAN, the context defines these values whereas in our system the user can give preferences between indexes.

6.4 The CARS System

A part of the CARS system has been described in [3]. Cases and problems are also defined in an object-oriented environment. Cases and problems are described by surface or abstract fuzzy attributes. Abstract attributes are computed using plausible rules inferences. This system doesn't provide any contextual help to the problem description and the case filtering step is not described. However, the originality of this system resides in the techniques employed in the selection of nearest cases. The selection uses fuzzy similarity measures between attributes based on fuzzy algebra:

$$S = s(X, Y) = M - |X - Y|$$

where X and Y are two fuzzy values and M the unit. Then the similarity S, which is a fuzzy set, is compared to reference fuzzy sets. Each reference fuzzy set R_i has a label such as `NO-MATCH`, `PARTIAL-MATCH`, `COMPLETE-MATCH`, and represents a matching degree. This comparison is done with the measure of inclusion $\frac{|S \cap R_i|}{|R_i|}$. This measure gives a degree of confidence to each matching degree. Finally, the matching degrees of each attribute are aggregated with their importance weight using different combination methods.

In comparison to FLORAN, there is no fuzzy pattern to build. This system does not allow for the expression of preferences in the problem description. Above all, fuzzy values can only be defined on a real domain because fuzzy algebra has not been defined in other types of domains.

7 Conclusion

To summarize, this paper has presented our case retrieval support system shell based on fuzzy logic, whose originality resides mainly in two points:

- First, we propose to use fuzzy logic and more precisely the possibility theory in a CBR context in order to tackle real problems: indeed, our system allows the expression of flexible data (imprecise and incomplete data, preferences) to represent the target and cases.
- Secondly, we have integrated and adapted classification techniques in our system for case filtering. Moreover our algorithm can be used at the same time to support a user, a novice in a domain, in describing the problem to be solved.

Moreover, adopting a cognitive approach in order to guarantee real CBR assistants, we wish to validate our system shell on a specific real application. With this application we also want to analyze more deeply the needs for target and case representation, and case retrieval in order to improve the user-machine cooperation at the knowledge level [23, 2].

However, we would like to point out some perspectives at both the knowledge and the symbol levels. For case and problem descriptions, it seems to us crucial to represent complex cases and then to manage similarity measures between them and the target. To do this, we want to first study the notion of *viewpoint*, which is very important in problem solving [8]. We would like to use techniques implemented in the TROPES object-oriented environment [16] to manage, in a more realistic way, static viewpoints and composite objects. For target description, we want to take into account *nuances* about data [17] ("very", "about", "often", and so on), and computed values using fuzzy rules as in CLASSIC, or using fuzzy algebra [6]. For case retrieval, the aggregation of possibility and necessity could use different types of combination like in the CARS system and in the SIMCAIR fuzzy database manager [1], depending on different semantics. Finally, instead of selecting the nearest cases, fuzzy sets could be used to represent the adaptable values in order to retrieve the best cases in terms of their adaptability [14].

References

1. V. Andrès. *Filtrage sémantique dans une base de données imprécises et incertaines : un système souple autorisant la formulation de requêtes composites pondérées.* PhD thesis, Paul Sabatier University of Toulouse: France, February 1989.
2. E. Armengol and E. Plaza. A Knowledge Level Model of Case-Based Reasoning. In *LNAI 837 - Topics in Case-Based Reasoning*, pages 53–64. Springer Verlag, University of Kaiserlautern: Germany, 1993.
3. P.P. Bonissone and S. Ayub. Similarity measures for case-based reasoning systems. In *Proc. of the 4th international conference on information processing and management of uncertainty in knowledge-based systems*, pages 161–172, Palma de Mallorca: Spain, July 1993.
4. M. Cayrol, H. Farreny, and H. Prade. Fuzzy pattern matching. *Kybernetes*, 11:103–116, 1982.
5. Cognitive Systems. *ReMind : Developer's Reference Manual.* 220-230 Commercial St., Boston, MA 02109, 1992.
6. D. Dubois and H. Prade. Fuzzy real algebra: some results. *Fuzzy Sets and Systems*, 2:327–348, 1979.
7. D. Dubois, H. Prade, and C. Testemale. Weighted fuzzy pattern matching. *Fuzzy sets and systems*, 28(3):313–331, 1988.
8. D. Galarreta and B. Trousse. Cooperation between activities in complex organizations : study directions in the design of space systems. In *Workshop on Artificial Intelligence and Knowledge-based systems for Space.*, ESTEC, Noordwijk: the Nederlands, May 1993.
9. C. Granger. An application of possibility theory to object recognition. *Fuzzy Sets ans Systems*, 28:351–362, 1988.
10. ILOG, 2 avenue Galliéni, BP 85, 94253 Gentilly Cedex FRANCE. *IlogTalk : Manuel de référence*, 1994.
11. M. Jaczynski. Similar case retrieval and use of fuzzy sets. Master's thesis, University of Nice: France (in French), September 1993.
12. P. Janssen, T. Schiex, and al. "représentation et traitement pratique de la flexibilité dans les problèmes sous contraintes". In *Proc.* 4^{e} *Journées Nationales PRC–GRD Intelligence Artificielle (in French).* Teknea, 1992.
13. J. Kolodner. Maintaining organization in a dynamic long-term memory. *Cognitive Science*, 7(4):243–280, 1983.
14. J. L. Kolodner. Judging which is the best case for a case-based reasoner. In Morgan Kaufman, editor, *DARPA 89:CBR Workshop*, pages 77–81, Holiday Inn, Pensacola Beach: Florida, May 1989.
15. M. Manago, K.D. Althoff, E. Auriol, R. Traphoner, S. Wess, N. Conruyt, and F. Maurer. Induction and reasonning from cases. In *EWCBR-93: first European Workshop on Case-Based Reasoning*, pages 313–318, November 1993.
16. O. Marino. Classification d'objets composites dans un système de représentation de connaissances multi-points de vue. In *8ème Congrès Reconnaissance des Formes et Intelligence Artificielle RFIA.*, Lyon: France, novembre 1991.
17. N. Mouaddib and O. Foucaut. Gestion des informations nuancées dans une base de connaissances en sciences naturelles. utilisation de la théorie de possibilité/nécessité. In *IV journées bases de données avancées*, Bénodet: France, 1988.
18. N. Mouaddib, O. Foucaut, P. Subtil, and D. Vilmin. FIRMS : un système orienté objet pour la représentation des informations nuancées, application aux sciences

économiques. In *Les applications des ensembles flous*, Nîmes: France, November 1992.
19. A. Newell. The knowledge level. *AI journal*, 18(2):82–127, 1982.
20. Enric Plaza and R. López de Mántaras. A case-based apprentice that learns from fuzzy examples. *Methodologies for Intelligent Systems*, 5:420–427, 1990.
21. S. Salotti. *"Filtrage flou et repreésentation centrée objet pour raisonner par analogie : le système FLORAN"*. PhD thesis, University of Paris XI, Orsay: France (in French), December 1992.
22. B. Trousse. *Cooperation between knowledge-based systems and CAD tools : the $\mathcal{ANAXAGORE}$ multi-agent environment*. PhD thesis, University of Nice: France (in French), 1989.
23. B. Trousse and W. Visser. Use of case-based reasoning techniques for intelligent computer-aided-design systems. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 3, pages 513–518, Palais de l'Europe - Le Touquet: France, October 1993.
24. W. Visser. Use of analogical relationships between design problem-solution representations: Exploitation at the action-execution and action-management levels of the activity. *Studia Psychologica*, 34(4-5):351–357, 1992.
25. W. Visser and B. Trousse. Reuse of designs : desperately seeking an interdisciplinary cognitive approach. In *Proc. of the IJCAI'93 workshop on Reuse of designs - an interdisciplinary approach,*, Chambery: France, 1993.
26. L.A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.
27. L.A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy sets and systems*, 1:3–28, 1978.