A TUPLE-ORIENTED ALGORITHM FOR DEDUCTION IN A FUZZY RELATIONAL DATABASE

IGNACIO J. BLANCO*
Dept. Languages and Computer Science, University of Almería
Carretera de Sacramento S/N, La Cañada de San Urbano, 04120, Almería, SPAIN
.iblanco@ual.es

MARIA J. MARTIN-BAUTISTA, OLGA PONS, and M. AMPARO VILA
Dept. Computer Science and Artificial Intelligence, University of Granada
Periodista Daniel Sacedo Aranda S/N, 18071, Granada, SPAIN
{mbautis,opc,vila}@decsai.ugr.es

Received October 2002
Revised January 2003

In this paper, we define the concept of generalized rule for making classical deduction with imprecise data, stored both data and rules in a fuzzy relational database represented in the GEFRED model. We propose a way of measuring the imprecision related to the calculation of a fact based on the matching degree of the facts in the database and the facts calculated while expanding the rules. In order to achieve this, classical algorithms for deduction are not appropriated and we propose the modifications that have to be applied on a classical tuple-oriented algorithm in order to design a new algorithm for deducing from imprecise data with generalized rules.

Keywords: fuzzy data, fuzzy deduction, fuzzy prolog, relational database

1. Introduction

The Relational Model\(^{11}\) was introduced as a powerful tool for knowledge representation. This model allows to store and to query many kind of data but has some drawbacks.

One of them is that only precise or well-defined data can be handled. To deal with imprecise information, many authors\(^{5,12,13,14,15,23,42}\) introduced extensions of the classical Relational Model from different points of view. Some of these approaches\(^{9,19,35,36,45}\) are based on the Fuzzy Linguistic Logic\(^{22,44}\), that is based on Fuzzy Sets Theory introduced by Zadeh\(^{43}\). Some of these fuzzy database models were integrated by Medina et al.\(^{31,32}\) within an extension of the classical Relational

*Corresponding author
Model called \textit{GEFRED}. This extended model was implemented in a commercial RDBMS by Galindo\textsuperscript{20} and completed by Blanco et al.\textsuperscript{7}.

Another drawback is that the Relational Model only can handle explicit data but these data might be used to infer new implicit data. One of the most used tools for making deductions is \textit{Logic} but it is necessary database to translate the database into a logical database using the Logic language. The first proposal for the logical representation of a relational database was introduced by Gallaire\textsuperscript{21}. From this point of view, all relations are considered predicates and all tuples are considered facts. Usually, there are two different mechanisms to carry out the deduction process in a logical database: \textit{Prolog}\textsuperscript{18}, oriented to the values of the attributes in a tuple, and \textit{Datalog}\textsuperscript{10}, oriented to sets of tuples.

Adding deductive capabilities to a database results in a system equivalent to a logic program. But logic programming ignores incomplete, imprecise or unknown data. To deal with this problem many authors\textsuperscript{1,2,17,18,24,25,26,27,28,33,34,37,38,40,41} have proposed the incorporation of uncertainty into logic programming, allowing uncertainty in data, relations, rules or combinations of solutions.

Baldwin et al.\textsuperscript{3,4} introduced a representation for imprecise data within the relational model and proposed an inference language for making deductions with this data, called FRIL. Baldwin relates a degree to every tuple in a relation that allows introducing imprecision in relations.

Pons et al.\textsuperscript{39} proposed the integration of the \textit{GEFRED} model and the logical representation of a fuzzy relational database. In this paper, the authors proposed an architecture for fuzzy deductive relational database using the conventional relational model as a basis. Taking this work as a starting point, Blanco et al.\textsuperscript{8} proposed the features of an algorithm for deduction in relational databases.

In this paper, we propose a modified tuple-oriented algorithm for deduction with the logical representation of a \textit{GEFRED} database.

The paper is organized as follows: in section 2, we introduce some concepts of the integrated model (\textit{GEFRED} and the logical model), that will be necessary for the definition of the generalized rule concept, introduced in section 3, and its relational representation, introduced in section 4. Using these definitions, we shall describe some important issues of the algorithm for deduction with imprecise data in section 5. Finally, in section 6, we will set out the conclusion and future research lines.

2. Previous models

2.1. Fuzzy Relational Database Model: GEFRED

This model extends the concept of domain for the representation of imprecise information and later uses this concept to extend the definition of relation.

\textbf{Definition 1} Let $D$ be a domain of discourse, $\tilde{P}(D)$ the set of possibility distributions defined on $D$ (those describing values UNKNOWN and UNDEFINED are
included). Let us consider the value NULL. A Generalized Fuzzy Domain $D_G$ is a set verifying:

$$D_G \subseteq \hat{P}(D) \cup \{\text{NULL}\}$$

This domain is able to contain values that can be:

- a single non numerical discrete value ($\text{Behavior}=\text{good}$, represented by the possibility distribution $\{1/\text{good}\}$),
- a single number ($\text{Age}=25$, represented by the possibility distribution $\{1/25\}$),
- a set of mutually exclusive non numerical discrete possible assignments ($\text{Quality}=\{\text{good, bad}\}$, represented by the possibility distribution $\{1/\text{good, 1/bad}\}$),
- a set of mutually exclusive numeric possible assignments ($\text{Age}=\{23, 24\}$, represented by the possibility distribution $\{1/23, 1/24\}$),
- a possibility distribution in a domain of non numerical discrete values ($\text{Behavior}=\{0.6/\text{bad, 0.7/normal}\}$),
- a possibility distribution in a numeric domain ($\text{Age}=\{0.7/23, 1/24, 0.8/25\}$, fuzzy numbers or linguistic labels),
- a real number belonging to $[0, 1]$ referring to degree of matching ($\text{Quality}=0.9$),
- an $\text{UNKNOWN}$ value with possibility distribution $\{1/d, \forall d \in D\}$ and meaning that attribute can have any value in the domain $D$,
- an $\text{UNDEFINED}$ value with possibility distribution $\{0/d, \forall d \in D\}$ and meaning that attribute can have no value in the domain $D$,
- a $\text{NULL}$ value with possibility distribution $\{1/\text{UNKNOWN, 1/UNDEFINED}\}$ and meaning that no information is known about attribute.

**Definition 2** A Generalized Fuzzy Relation is a pair of sets $(H, B)$ with the following definition:

- $H$ is the set named head and defining the structure of the relation as a set of triples (attribute, generalized fuzzy domain, compatibility degree) where the last is optional:
  $$H = \{(A_{G_1}, D_{G_1}, [c_{A_{G_1}}]), \ldots , (A_{G_n}, D_{G_n}, [c_{A_{G_n}}])\}$$

- $B$ is the set named body that describes the content of the relation as a set of triples attribute-value-degree where the last one is optional:
  $$B = \{(A_{G_1}, \tilde{d}_{i_1}, [c_{i_1}]), \ldots , (A_{G_n}, \tilde{d}_{i_n}, [c_{i_n}])\}$$
In order to compare values in the same generalized fuzzy domain, the model introduces the concept of generalized fuzzy comparator based on the definition of the concept of extended fuzzy comparator.

**Definition 3** Let $U$ be the domain of discourse being considered. An extended fuzzy comparator $\theta$ is a fuzzy relation defined on $U$ as follows:

$$
\theta : U \times U \rightarrow [0, 1] \\
\theta(u_i, u_j) \mapsto a
$$

**Definition 4** Let $U$ be a domain of discourse, $D$ a fuzzy domain on $U$ and $\theta$ an extended comparator defined on $U$. Let $\Theta^\theta$ be the function with the following definition:

$$
\Theta^\theta : D \times D \rightarrow [0, 1] \\
\Theta^\theta(\tilde{d}_1, \tilde{d}_2) \mapsto [0, 1]
$$

$\Theta^\theta$ is named a generalized fuzzy comparator on $D$ and generated by the extended comparator $\theta$ if it verifies:

$$
\Theta^\theta(\tilde{d}_1, \tilde{d}_2) = \theta(d_1, d_2), \forall d_1, d_2 \in U
$$

where $\tilde{d}_1$ and $\tilde{d}_2$ are the possibility distributions $\{1/d_1\}$ and $\{1/d_2\}$ defined on the values $d_1$ and $d_2$, respectively.

### 2.2. Logical Model for Relational Databases

The theory of databases is strongly based on Logic, specially when building queries and when defining views and integrity constraints.

In deductive databases, two types of relations can be defined:

- **Extensional relation** in the sense of the classical Relational Model, that is, a pair of sets $(R, r)$ where $R$ is a set of attributes defining the structure of the relation, and $r$ is a set of tuples defining the content of the relation.

- **Intensional relation** is a pair of sets $(R, I)$ where $R$ is a set of attributes defining the structure of the relation and $I$ is a set of logical formulas, called instance generator, to obtain the content of the relation.

The rules in the instance generator $I$ have different structure in both the classical and the fuzzy cases.

A proposal for the rule scheme in the classical deductive database model was introduced by Pons et al. and Blanco et al. We establish that every rule in the set $I$ of a classical deductive database has the following structure:

$$
P(X_1, X_2, \ldots, X_n) : - \\
Q_1(Y_{1,1}, Y_{1,2}, \ldots, Y_{1,n_1}) \wedge \ldots \wedge \\
\wedge \ldots \wedge Q_m(Y_{m,1}, Y_{m,2}, \ldots, Y_{m,n_m})
$$

(1)
The predicate \( P \), whose facts are being calculated, is called head of the rule and the logical formula at the right of \(-\) is called body of the rule. We impose the following constraints on a rule to be considered correct for the instance generator:

(i) every variable in the head of the rule appears at least once in the body of the rule, and
(ii) every variable in the body of the rule appears in the predicate of the head of the rule or in any other predicate of the body of the rule.

3. Generalizing Prolog-type rules for making deduction with imprecise data

In this section, we apply the concepts of the deductive database model to the fuzzy relational model GEFRED.

First, we define some concepts as extensions of the concepts shown in the previous section.

**Definition 5** An extensional fuzzy relation is a Generalized Fuzzy Relation from the GEFRED Model point of view (definition 2), that is, a pair of sets \((H, B)\) where \(H\) corresponds to the scheme and \(B\) corresponds to the instance or body of the relation.

For example, let us suppose the following medical relations:

- **MedicalRecord** represents the medical records of the patient and it has the following attributes:
  - \(N\text{MedRec}\): a number to identify the concrete medical record and
  - \(Age\): an imprecise attribute having values in a generalized fuzzy domain defined on \([1 - 100]\). Some linguistic concepts, such as young, middle and old, have been defined in order to refer to some pre-defined possibility distributions.

- **Risks** represents the probability of suffering an illness depending on the age; this relation has the following attributes:
  - \(Age\): attribute defined in the same way as the previous relation one,
  - \(Illness\): a string representing the illness the patient can suffer from,
  - \(Probability\): an imprecise attribute having values in another generalized fuzzy domain defined on \([1 - 100]\) too. Other linguistic concepts, such as high, medium and low, have been defined on this domain.

A graphic representation of the head and the body of these relations is shown in figure 1.

**Definition 6** An intensional fuzzy relation is a pair of sets \((H, I)\) where:
• *H* is the set named head and defined as shown in the definition 2 and

• *I* is the set named instance generator, that is constituted by the set of rules that every fact of the associated predicate has to verify.

For example, let us suppose an intensional fuzzy relation *MapSufferFrom* whose content depends on the content of relations *MedicalRecord* and *Risks*. The head of *MapSufferFrom* has the following attributes:

• **NMeredRec**: attribute referred to that belonging to the relation *MedicalRecord*,

• **Illness**: attribute referred to that belonging to the relation *Risks*, and

• **Probability**: attribute referred to that belonging to the relation *Risks*.

Now, we are going to generalize the definition of rule in the instance generator *I* so that classical and flexible rules can be represented.

Let us use the classical expression of a rule shown in (1). We consider that a variable appears in two different predicates if one of these situations occurs:

(i) \( P(..., X_{i}, ...) \) and \( Q(..., X_{i}, ...) \) are in the rule,

(ii) \( Q(..., X_{i}, ...) \) and \( Q(..., X_{j}, ...) \) with \( i \neq j \) are in the rule,

(iii) \( P(..., X_{i}, ...), Q(..., Y_{j,k}, ...) \) and \( (X_{i} = Y_{j,k}) \) are in the rule, or

(iv) \( Q(..., Y_{j,k}, ...), Q(..., Y_{l,p}, ...) \) and \( (Y_{j,k} = Y_{l,p}) \) with \( l \neq p \) are in the rule.

It is obvious that situations 1 and 2 can be rewritten so that only the last two situations have to be observed. The constraint \( (A = B) \) with \( A \) and \( B \) variables is called link between \( A \) and \( B \).

The structure of the predicates in the classical or the fuzzy relational model is not different because this structure does not describe the domains or the relation among the domains.

In the classical case, \( (A = B) \) is evaluated to true or false. But when working on generalized fuzzy domains, the comparison \( (A = B) \) has uncountable truth values between true and false (or in \([0, 1]\)). The equality operator \( = \) does not calculate whether the values of two variables are exactly equal but the similarity or nearness between them. The same occurs with the other comparison operators. We represent all of them with the symbol \( \Theta^e \).
From the fuzzy point of view, it could be interesting to ignore pairs of values whose similarity degree is under a given threshold \( \tau \), so we consider a condition is held if the similarity degree is greater or equal than \( \tau \). This situation is represented with a new condition \( (A =_\tau B) \) where \( \tau \) is the threshold for the condition to be fulfilled, so the following equivalence is satisfied:

\[
(A =_\tau B) \equiv (\Theta^= (A, B) \geq \tau)
\]

In the Fuzzy Set theory, if \( D \) is a domain of discourse and \( \hat{P}(D) \) is the set of all possibility distributions that can be defined on \( U \) then, it is verified that \( \hat{d} \in \hat{P}(D) \) where \( \hat{d} \) is the fuzzy set described by the distribution \( \{1/d\} \) and it is called singleton. Every operator \( =_\tau \) verify that \( \forall \tau \in [0, 1], \forall d_1, d_2 \in D, \, (d_1 \neq d_2) \text{ then } d_1 =_\tau d_2 \text{ is evaluated to false.} \) With this constraint, all comparisons in the classical model are possible in the fuzzy model.

If \( A \) and \( B \) are variables having values in classical domains \( D_A \) and \( D_B \) then there exist two variables \( \hat{A} \) and \( \hat{B} \) having values in fuzzy domains \( \hat{D}_A \) and \( \hat{D}_B \) such as:

\[
\begin{align*}
\hat{D}_A &= \{1/d \mid d \in D_A\} \\
\hat{D}_B &= \{1/d \mid d \in D_B\}
\end{align*}
\]

satisfying:

\[ (A = B) \Leftrightarrow (\hat{A} = \hat{B}) \]

With all these considerations, we can establish the structure for a generalized rule that can be applied in the classical and the fuzzy model.

**Definition 7** Let be:

- \( P \) a \( n \)-ary predicate and \( Q_{i, i} \in \{1, \ldots, m\} \) \( n_i \)-ary predicates.
- \( X_i, i \in \{1, \ldots, n\} \) and \( Y_{j, k}, j \in \{1, \ldots, m\}, k \in \{1, \ldots, n_j\} \) variables taking their values from fuzzy generalized domains \( D_{X_i, i} \in \{1, \ldots, n\} \) and \( Y_{j, k}, j \in \{1, \ldots, m\}, k \in \{1, \ldots, n_j\} \).

We define a generalized rule for deduction with fuzzy data, or simply generalized rule, a rule with the following structure:

\[
P(X_1, X_2, \ldots, X_n) : \quad \neg Q_1(Y_{1, 1}, Y_{1, 2}, \ldots, Y_{1, n_1}) \land \ldots \land \neg Q_m(Y_{m, 1}, Y_{m, 2}, \ldots, Y_{m, n_m}) \land \Psi
\]

where \( \Psi \) is a formula:

\[
\begin{align*}
\Psi &\equiv \land (X_i =_{\alpha_{i, j, k}} Y_{j, k}) \land \\
&\land (Y_{j, k} =_{\beta_{j, k, l, p}} Y_{l, p}) \land \\
&\land (\phi_{j, k, l, p}(\Theta_{j, k, i, p}(Y_{j, k}, Y_{l, p}), Y_{j, k, l, p}))
\end{align*}
\]

and where:
- $\bigwedge A_i$ represents $A_1 \land A_2 \land \ldots \land A_n$.
- conditions $(X_i = \alpha_{i,k}, Y_{j,k})$ refer to links of type 3 between variables,
- conditions $(Y_{j,k} = \beta_{j,k,i,p}, Y_{i,p})$ refer to links of type 4 between variables,
- expressions $\phi_{j,k,i,p}(\Theta_{j,k,i,p}^\theta(Y_{j,k}, Y_{i,p}), \gamma_{j,k,i,p})$ refer to comparisons explicitly introduced by the user to constraint the rule meaning with:
  - $\phi_{j,k,i,p}$ the function to determine if the satisfaction degree for the comparison is greater than the threshold,
  - $\Theta_{j,k,i,p}^\theta$ is a generalized fuzzy comparator defined on the classical comparator $\theta$
  - $\gamma_{j,k,i,p}$ is the minimum degree for the comparison to be evaluated to true.
- every variable in the head of the rule is linked to one or more variables in the body of the rule (situation 3), and
- every variable in the body of the rule is linked to a variable in the head or in the body of the rule.

This definition allows to soften the matching between variables in the body and the head of the rule. Subsequently, this softening implies a relaxation of the concept of consistency because non exact facts are considered consistent facts.

The result obtained when calculating facts for a given predicate $P(X_1, \ldots, X_n)$, is a set of facts $P(a_{i,1}, \ldots, a_{i,n})$. Some of them are exact and some others are near to those exact ones. We can give a definition for the “distance” between the exact results and the approximated ones, and this definition can be seen as a measure for the flexible consistency.

To achieve this, the result set $\{P(a_{i,1}, \ldots, a_{i,n})\}$ can be re-structured as a set of pairs $\{(P(a_{i,1}, \ldots, a_{i,n}), \gamma_i)\}$ where $\gamma_i$ is the nearness degree to an exact solution. In this set, exact solutions are characterized by $\gamma_i = 1$. In order to unify the notation, we use the following translation:

$$(P(X_1, \ldots, X_n), \gamma_p) \equiv \tilde{P}(X_1, \ldots, X_n, \gamma_p)$$

The translation applied to the expression (2) gives rise to a new definition of rule.

**Definition 8** Let us consider a generalized rule as shown in expressions (2) and (3). A generalized fuzzy rule with matching degree is a rule with the following structure:

$$\tilde{P}(X_1, X_2, \ldots, X_n, \gamma_{\tilde{P}}) : - Q_1(Y_{1,1}, Y_{1,2}, \ldots, Y_{1,n_1}, \gamma_{Q_1}) \land \ldots \land \\
\land Q_m(Y_{m,1}, Y_{m,2}, \ldots, Y_{m,n_m}, \gamma_{Q_m}) \land \Psi$$

where $\Psi$ is a formula with the structure shown in the expression (3) and $\gamma_{\tilde{P}}$ is a matching function that considers the satisfaction degree of the conditions in the
formula Ψ and the satisfaction degree of the facts calculated in the instantiation of the rule.

The degree γ_p refers to the matching degree between the calculated fact and an exact fact of the predicate. This degree depends on the matching degree of every fact used for the calculation in the body of the rule and on the satisfaction degree of every flexible condition appearing in the formula Ψ. Therefore, the expression for γ_p is:

\[ \Phi(\Theta^{\text{w}}_{i,j,k}(X_{i1},Y_{i,k}), \ldots, \Theta^{\text{w}}_{j,k,l,p}(Y_{j,k},Y_{l,p}), \ldots, \Theta^{\text{f}}_{j,k,l,p}(Y_{j,k},Y_{l,p}), \ldots, \gamma_{Q1}, \ldots, \gamma_{Qm}) \]

All these degrees are calculated for predicates and conditions joint by the operator ∧. We propose to use an aggregation function of the fuzzy logic such as the minimum function. It allows to calculate an only matching degree for every fact.

If the rule has l conditions and m predicates, the matching function for this rule has the following structure:

\[ \Phi(c_1, \ldots, c_l, \gamma_1, \ldots, \gamma_m) \]

and if the distributive property of the minimum function is applied to the expansion of this expression, the matching function is the following one:

\[ \Phi(c_1, \ldots, c_l, \gamma_1, \ldots, \gamma_m) = \text{Min}(\text{Min}(c_1, \ldots, \text{Min}(c_{l-1}, c_l), \ldots)), \text{Min}(\gamma_1, \ldots, \text{Min}(\gamma_{m-1}, \gamma_m))) \]

(5)

In the previous example, a rule for calculating the content of the relation MaySufferFrom is shown in expression (6).

\[ \text{MaySufferFrom}(X,Y,Z_{CR}, \text{Min}(\Theta^=(T_1,T_2), \Theta^=(Z_{RP}, Z_{CR}))) \leftarrow \text{MedicalRecord}(X,T_1) \land \text{Risks}(T_2, Y, Z_{RP}) \land (T_1 \neq T_2) \land \]

(6)

\[ \land (Z_{RP} = a, Z_{CR}) \]

3.1. Combining results from different rules

Usually, a predicate \( \bar{P} \) is defined by a disjunction of several rules \( \bar{P}_i \) as follows

\[ \bar{P}(X_1,X_2,\ldots,X_n,\gamma_p) \leftarrow \bar{P}_1(X_1,X_2,\ldots,X_n,\gamma_{\bar{P}_1}) \lor \ldots \lor \bar{P}_r(X_1,X_2,\ldots,X_n,\gamma_{\bar{P}_r}) \]

When rules are not exclusive then a fact can result from several rules with possibly different satisfaction degrees. These satisfaction degrees have to be combined by a function \( \Omega \).

With this consideration, for every fact \( \bar{P}_i(a_1,a_2,\ldots,a_n,\gamma_i) \) resulting from a rule calculus we can build a fact of the predicate \( \bar{P} \) with the structure shown in (7)

\[ \bar{P}(a_1,a_2,\ldots,a_n,\Omega(0,\ldots,\omega_i,\ldots,0)) \]

(7)
where $\Omega(\omega_1, \ldots, \omega_n)$ is a function for the combination of the degrees and where:

$$\omega_i = \begin{cases} 
\gamma_i & \text{if the fact results from the rule } \tilde{P}_i \\
0 & \text{otherwise}
\end{cases}$$

When a fact results from more than one rule with several satisfaction degrees, it satisfies the predicate at the higher degree. So it is necessary a function $\Omega$ that is related to the disjunction operator. We propose to use an aggregation function of the fuzzy logic such as the maximum as shown in (8).

$$\Omega(\omega_1, \omega_2, \ldots, \omega_{n-2}, \omega_{n-1}, \omega_n) = \text{Max}(\omega_1, \text{Max}(\omega_2, \text{Max}(\ldots, \\
\text{Max}(\omega_{n-1}, \omega_n) \ldots)))$$

### 4. Relational representation of generalized rules

This mechanism has been introduced to make deductions with precise and imprecise data stored in relational structures. In order to maintain the consistency of this proposal, we use a relational representation for the logical information based on the FREDDI interface, introduced by Medina et al. This set of relations allows to store the definition of a predicate as a disjunction of one or several rules. Each one of them is defined as a conjunction of predicates and comparisons.

As seen in section 3, predicates are the same in the classical and generalized rules except from the matching degree that has not to be stored in database. But the set of comparisons have to be modified so that imprecise matching between variables can be represented. With this consideration, the relational structure is modified as shown in figure 2.

![Relational structure to store logical information](image)

Figure 2: Relational structure to store logical information

The modification of the relation Condition_description allows the representation of comparisons based on imprecise operators. In this example we have used the imprecise operators detailed by Medina et al. and Galindo.

### 5. An algorithm based on the Prolog engine to work with generalized rules

In order to design an algorithm to make deductions with imprecise data, we have to
examine how a classical Prolog-based algorithm operates. This allows to establish the points where classical data are used and to determine what modifications are necessary for deducing with imprecise data.

5.1. An overview of the classical mechanism

Blanco et al.\(^6\) proposed an algorithm to be applied on classical rules structured as shown in the expression (1). This algorithm is based on the Prolog deduction mechanism.

The basic operation of this proposal is the construction of a generic tuple for every rule as shown in figure 3, and it applies every predicate of the rule producing several instances of this generic tuple.

\[
\begin{array}{cccccccc}
X_1 & \ldots & X_n & Y_{1,1} & \ldots & Y_{1,a_1} & \ldots & Y_{m,1} & \ldots & Y_{m,a_m} \\
\uparrow & & & \uparrow & & & \uparrow & & & \uparrow \\
Q_1 & \cdots & Q_m
\end{array}
\]

Figure 3: Generic tuple of an intensional relation

Let us assume an intensional predicate \(P\) defined by the following set of rules:

\[
P(X,Y) : -Q(X,Z) \land R(Z,Y) \\
R(X,Y) : -S(Y,X)
\] (9)

The mechanism operates as shown in figure 4.

\[
\begin{array}{cccccccc}
X & Y & Z \\
\uparrow & \uparrow & \uparrow \\
s_1 & s_2 & s_3 \\
\cdots & \cdots & \cdots \\
s_n & s_2 & s_3 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
s_1 & Y_{1,1} & s_1 \\
\cdots & \cdots & \cdots \\
s_n & Y_{1,a_1} & s_1 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
s_1 & Y_{2,1} & s_2 \\
\cdots & \cdots & \cdots \\
s_n & Y_{2,a_2} & s_2 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
s_1 & Y_{3,1} & s_3 \\
\cdots & \cdots & \cdots \\
s_n & Y_{3,a_3} & s_3 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
s_1 & Y_{4,1} & s_1 \\
\cdots & \cdots & \cdots \\
s_n & Y_{4,a_n} & s_1 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
s_1 & Y_{5,1} & s_2 \\
\cdots & \cdots & \cdots \\
s_n & Y_{5,a_n} & s_2 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
s_1 & Y_{6,1} & s_3 \\
\cdots & \cdots & \cdots \\
s_n & Y_{6,a_n} & s_3 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
s_1 & Y_{7,1} & s_1 \\
\cdots & \cdots & \cdots \\
s_n & Y_{7,a_n} & s_1 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
s_1 & Y_{8,1} & s_2 \\
\cdots & \cdots & \cdots \\
s_n & Y_{8,a_n} & s_2 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
s_1 & Y_{9,1} & s_3 \\
\cdots & \cdots & \cdots \\
s_n & Y_{9,a_n} & s_3 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
s_1 & Y_{10,1} & s_1 \\
\cdots & \cdots & \cdots \\
s_n & Y_{10,a_n} & s_1 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
s_1 & Y_{11,1} & s_2 \\
\cdots & \cdots & \cdots \\
s_n & Y_{11,a_n} & s_2 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
s_1 & Y_{12,1} & s_3 \\
\cdots & \cdots & \cdots \\
s_n & Y_{12,a_n} & s_3 \\
\end{array}
\]

Figure 4: Deduction mechanism

The steps for this operation are summarized in the following list:

(i) The predicate \(Q\) is applied to the generic tuple and this operation generates a set of instances of this tuple as a result of querying the extensional relation associated with the extensional predicate \(Q\).

(ii) The intensional predicate \(R\) is applied to every instance \(i\).
(iii) If \( R \) is an intensional predicate, it produces the calculation of the facts satisfying the predicate \( R \) with some variables having concrete values \( z_i \). The predicate \( R \) is based on an extensional predicate \( S \) and the application of this one (querying an extensional relation) generates a set of instances for every input tuple.

(iv) The set of instances of every input tuple modifies the tuple of \( P \), which generates new instances for this tuple.

(v) Every tuple gives rise to a fact for the predicate \( P \).

5.2. The order when applying predicates

In the step (1), predicate \( Q \) is applied and then values for the variables \( X \) and \( Z \) are fixed. In steps (2), \( R \) is applied and then values for the variable \( Y \) are fixed. This situation can be represented with the graph shown in figure 5.

\[
\emptyset \xrightarrow{Q(X,Z)} \{X,Z\} \xrightarrow{R(Z,Y)} \{X,Z,Y\}
\]

Figure 5: Graph of variables marking

If we extend the rules in the expression (9) by applying the procedure seen in the previous section, the extended rules are the following:

\[
\begin{align*}
\tilde{P}(X_P, Y_P) & : \neg \tilde{Q}(X_{\tilde{Q}}, Z_{\tilde{Q}}) \land \\
& \land \tilde{R}(Z_{\tilde{R}}, Y_{\tilde{R}}) \land (Z_{\tilde{Q}} = Z_{\tilde{R}}) \land \\
& \land (X_P = X_{\tilde{Q}}) \land (Y_P = Y_{\tilde{R}}) \\
\tilde{R}(X_{\tilde{R}}, Y_{\tilde{R}}) & : \neg \tilde{S}(Y_{\tilde{S}}, X_{\tilde{S}}) \land \\
& \land (X_{\tilde{R}} = X_{\tilde{S}}) \land (Y_{\tilde{R}} = Y_{\tilde{S}})
\end{align*}
\]

(10)

Let us notice that variables \( X_{\tilde{R}} \) and \( Y_{\tilde{R}} \) in the first and second rules are not the same. The graph shown in the figure 5 should be modified as figure 6 shows.

\[
\emptyset \xrightarrow{\tilde{Q}(X_{\tilde{Q}}, Z_{\tilde{Q}})} \{X_{\tilde{Q}}, Z_{\tilde{Q}}\} \xrightarrow{\tilde{R}(Z_{\tilde{R}}, Y_{\tilde{R}})} \{X_{\tilde{Q}}, Z_{\tilde{Q}}, Z_{\tilde{R}}, Y_{\tilde{R}}\} \xrightarrow{(Z_{\tilde{Q}} = Z_{\tilde{R}})} \{X_{\tilde{Q}}, Z_{\tilde{Q}}, Z_{\tilde{R}}, Y_{\tilde{R}}, X_{\tilde{P}}\} \xrightarrow{(X_P = X_{\tilde{Q}})} \{X_{\tilde{Q}}, Z_{\tilde{Q}}, Z_{\tilde{R}}, Y_{\tilde{R}}, X_{\tilde{P}}, Y_{\tilde{P}}\} \xrightarrow{(Y_P = Y_{\tilde{R}})} \{X_{\tilde{Q}}, Z_{\tilde{Q}}, Z_{\tilde{R}}, Y_{\tilde{R}}, X_{\tilde{P}}, Y_{\tilde{P}}\}
\]

Figure 6: Extended graph of variables marking

In this figure, condition evaluations are postponed to be done at the end of the rule and this increases considerably the number of tuples used for calculation. If the execution is re-ordered from the beginning, the mechanism will be more efficient. A possible re-ordering of the execution is shown in figure 7.
The arcs in the graph of the figure 5 are referred to:

- a query to the extensional relation associated to the predicate, if the predicate being applied is \( Q \), or

- a calculation of the facts associated to the predicate and a query related to the results obtained by the application of rules in the instance generator, in the case of the predicate \( R \).

Classical query languages are insufficient when working on flexible values because they do not include flexible conditions. On the basis of the GEFRED model, it introduced the relational structure for the representation of flexible information and Galindo developed a query language for data stored in this structure. This language is an extension of SQL called FSQL.

The application of every arc of the graph in figure 7 is related to a query in FSQL taking into account the considerations made before about extensional and intensional predicates. If an arc contains more than one predicate or condition, they are applied in an only condition composed by two of them joint by the conjunction operator.

Summarizing, before applying a rule, it is necessary to determine the order of application for every predicate and condition so the convergence of the mechanism can be assured.

This order depends on the variables that have been fixed in the requested query. If we use the rule shown in expression (6), the calculation of facts coupling with \( \text{MaySufferFrom}(a, \ldots) \) is carried out by applying predicates and conditions in the rule in the order shown in figure 8.

\[
\begin{align*}
\{ \& \} & \xrightarrow{\text{MedicalRecord}(X,T_1)} \{ \& ,T_1 \} \\
\{ \& ,T_1 \} & \xrightarrow{\text{Risks}(T_2,Y,Z_{RP})} \{ \& ,T_1 ,T_2 ,Y,Z_{RP} \} \\
\{ \& ,T_1 ,T_2 ,Y,Z_{RP} \} & \xrightarrow{\text{Z}_{RP}=a\text{Z}_{CR}} \{ \& ,T_1 ,T_2 ,Y,Z_{RP},Z_{CR} \}
\end{align*}
\]

Figure 8: Graph of variables marking for \( \text{MaySufferFrom}(a, \ldots) \)
In another case, if facts coupling with \textit{MaySufferFrom}(2,\leq 60,\ldots) are being calculated then the order of application is shown in figure 9.

\[
\begin{align*}
(Z_{RP} = Z_{CR}) \\
(X, Z_{CR}) \xrightarrow{\text{MedicalRecord}(X,T_1)} \{X, Z_{CR}, T_1\} \xrightarrow{T_1 = T_2} \{X, T_1, T_2, Y, Z_{RP}, Z_{CR}\} \\
\text{Risks}(T_2, Y, Z_{RP})
\end{align*}
\]

Figure 9: Graph of variables marking for \textit{MaySufferFrom}(a,\ldots,b,\ldots)

5.3. An overview for the generalized algorithm

The algorithm presented by Blanco et al.\textsuperscript{6} uses a relation for each rule defining the calculated predicate. A relation stores all copies of the generic tuple associated to the rule.

A predicate expansion initializes all relations associated to the predicate rules with a generic tuple. These relations are managed as stacks, so only one tuple is the active one at every moment. In each step, the algorithm selects the first predicate in the rule that has not been applied yet to the active tuple. This is possible because predicates in a rule have common variables and the application of predicates reduces the number of tuples to be considered.

This classical proposal is tuple-oriented and each step results on the application of a predicate. Furthermore, variables appearing in two predicates are implicitly managed by the algorithm. The modification of the algorithm preserves the feature of the tuple orientation. But, as seen in section 3, each variable appearing in two different predicates is adapted to the generalized rule form by being replaced with two different variables (a different variable in each predicate) and a comparison linking them. This will be the new second feature of our extension.

The original proposal applies a predicate to the active tuple in each step of the algorithm. Originally, all predicates have the same "weight" in a classical Prolog-based mechanism, so the order in applying predicates is not so important. And a comparison is a special kind of binary predicate.

The main point of taking a comparison apart from the rest of the rule is the problem of the Universe, that is, a variable in an isolated comparison can take an infinite number of values. The original proposal faced up with the problem by delaying the comparison application until both variables had been instantiated. This was possible because of the same priority of predicates and comparisons, so last ones could be postponed and the resulting set would not be changed.

However, this postponing concerns to the speed in obtaining a resulting set of tuples and the number of tuples that have to be stored. We need to assure that only relevant combinations are considered (optimization of speed) and stored (optimization of space). From an informal point of view, we can conclude that the original proposal applied a predicate and a comparison when applying a predicate with an instantiated variable. So classical algorithm applies predicate blocks.
Definition 9 Let $\tilde{P}(X_1, X_2, \ldots, X_n, \gamma_P)$ be a predicate defined by a generalized rule as shown in expression (4). A Predicate Block $PB_j$ is a formula inside the rule with the following structure:

$$PB_j = \begin{cases} C_{\tilde{Q}_j} \land \tilde{Q}_j(Y_{j,1}, Y_{j,2}, \ldots, Y_{j,n_j}, \gamma_{\tilde{Q}_j}) \\ \tilde{Q}_j(Y_{j,1}, Y_{j,2}, \ldots, Y_{j,n_j}, \gamma_{\tilde{Q}_j}) \\ \text{a comparison in } \Psi \end{cases}$$

(11)

where $\tilde{Q}_j$ is a predicate in the rule, $C_{\tilde{Q}_j} = \land C_r$ is a sub-formula inside the formula $\Psi$, shown in the expression (3), and every $C_r$ can be one of the following:

- $(X_i = a_{i,j,k} Y_{j,k})$ with $Y_{j,k}$ appearing in $\tilde{Q}_j$, and $X_i$ instantiated,
- $(Y_{j,k} = b_{j,k,l,p} Y_{l,p})$ with $Y_{j,k}$ or $Y_{l,p}$ appearing in $\tilde{Q}_j$, and the other variable instantiated, or
- $\phi_{j,k,l,p}(\Theta_{j,k,l,p}^d(Y_{j,k}, Y_{l,p}, \gamma_{j,k,l,p}))$ with $Y_{j,k}$ or $Y_{l,p}$ appearing in $\tilde{Q}_j$, and the other one instantiated.

In other words, a predicate block is a sub-formula containing a predicate and all comparisons having a common free variable with the predicate and the other variable with an already fixed value, or a single predicate or comparison. Every arc in figures 8 and 9 represents a predicate block.

In the classical approach of the algorithm, the order in the application of the predicates could be changed if necessary. In our extension, this order has to be previously calculated, as shown in 7, so every arc in the graph is translated to a predicate block and the order of the arcs represents the order in which predicate blocks have to be applied.

The structure of relations associated to the rules has to be modified, because more information has to be stored. In every rule we have to store the values for all variables appearing in the head and the body, and the satisfaction degree of every predicate.

The generic tuple of a relation have the following structure:

<table>
<thead>
<tr>
<th>IDSET</th>
<th>VAR1</th>
<th>\ldots</th>
<th>VARn</th>
<th>PHI1</th>
<th>\ldots</th>
<th>PHIm</th>
<th>NEG</th>
<th>\ldots</th>
</tr>
</thead>
<tbody>
<tr>
<td>\ldots</td>
<td>INST</td>
<td>APPL</td>
<td>TABLEP</td>
<td>IDSETP</td>
<td>CTRL</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

where the VARi attributes store the values of the variables in the head and the body of the rule, the PHIj attributes store the satisfaction degree of every predicate in the body of the rule, the APPL attribute store the number of predicate blocks applied to the tuple, and the others preserve their meaning.

With these modifications, an overview of the extended algorithm operates on a values set as following:

1. If attribute CTRL is 0, then it is modified (there are updated copies of it) and the set must be deleted.
2. If attribute CTRL is 1, then it is a valid values set. If there are predicate blocks that have not been applied (predicate blocks of the rule minus APPL attribute is not zero), the next predicate block is applied:

(a) If the active stack relation is intensional and has not been previously calculated, the mechanism determines the predicate of the next predicate block to be applied and its type. If the predicate in the predicate block is:

i. extensional: then this predicate block can be applied by translation into a flexible query with constraints based on the fixed values of the variables. Values in every tuple of the resulting relation are promoted to a copy of the tuple that is being calculated. Before beginning a new tuple processing, the status of the calculated tuple is modified.

ii. intensional: then this predicate block can not be applied until the predicate involved has been calculated. To do it, a modified empty values set is inserted into all stack relations associated to the intensional predicate rules and the mechanism begins the processing of the new inserted tuples. Once the intensional predicate has been calculated, the predicate block containing it can be applied as a flexible query.

iii. comparison: then comparison (binary predicate) is evaluated. The set must be marked as modified if it does not satisfy the comparison (CTRL = 0).

The set APPL attribute is increased because a new predicate has been expanded and a new predicate block has been applied. Finally, the next predicate stack relation is set to active.

(b) If the active stack relation is previously calculated as extensional or intensional, the mechanism increases the APPL attribute in the active values set (APPL = 1), obtains data from the extensional or intensional pre-calculated relation and inserts a copy of the active values set with the values modified for each tuple in the query result. The active values set is marked to modified (CTRL = 0).

If there are no predicate blocks to be applied, two different situations can arise:

(a) If there are no fixed-value variables, then no value combination satisfies the rule when all predicate blocks have been applied. Therefore the set can be deleted.

(b) If there are no free variables, the separator for this expansion is used to promote the active set values to a copy of the value set that expands the predicate. Therefore this expanded set is marked to modified (CTRL = 0) and the active set is deleted.
3. If the values set CTRL attribute is 2 (separator), then all these expansion sets have been processed and we can mark to activate the set that made this expansion.

If the values set is not modified and the active stack relation is extensional (PRED), then the data are obtained from the extensional relation associated to it and a copy for each combination of the result is introduced in the stack relation. These data are those which satisfy the extensional predicate. The values set that expands the predicate is marked to modified. A predicate has been applied to all copies in the extensional stack relation so the APPL increases by a unit.

5.4. An example

In order to illustrate the operation of the proposed mechanism, we execute it on the next intensional and extensional predicates:

- the intensional predicate *MaySufferFrom*, described by a variation of the rule shown in expression (6) with the value $\beta = 0.4$,

$$
\text{MaySufferFrom}(X, Y, \text{Min}(\Theta = (T_1, T_2), \gamma_{\text{MedicalRecord}}, \gamma_{\text{Risks}})) \leftarrow \\
\text{MedicalRecord}(X, T_1) \land \text{Risks}(T_2, Y) \land (T_1 =_\beta T_2) 
$$

(12)

- the extensional predicates *MedicalRecord* and *Risks*, shown in figure 1.

A calculation of all facts satisfying the predicate *MaySufferFrom* would be done in two steps applied to every tuple. That is, the rule defining the predicate can be re-ordered in two parts (predicate blocks) being:

- $PB_1 = \text{MedicalRecord}(X, T_1)$, and
- $PB_2 = (T_1 =_\beta T_2) \land \text{Risks}(T_2, Y)$.

At first, the mechanism works on a generic tuple. When applying $PB_1$ to this tuple (common part of all tuples has been ignored), it results in the next set of tuples:

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>T_1</th>
<th>T_2</th>
<th>PHI_1</th>
<th>PHI_2</th>
<th>PHI_3</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Y</td>
<td>54-56</td>
<td>T_2</td>
<td>1</td>
<td>PHI_2</td>
<td>PHI_3</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>Y</td>
<td>$\approx$ 35</td>
<td>T_2</td>
<td>1</td>
<td>PHI_2</td>
<td>PHI_3</td>
<td>...</td>
</tr>
<tr>
<td>3</td>
<td>Y</td>
<td>65</td>
<td>T_2</td>
<td>1</td>
<td>PHI_2</td>
<td>PHI_3</td>
<td>...</td>
</tr>
</tbody>
</table>

The next tuple to be calculated is the last one (the one above the double line), so $PB_2$ is applied and values $T_2$, $PHI_2$ and $PHI_3$ are fixed if the flexible query results in a set of tuples.

| 1 | Y | [54-56] | T_2 | 1 | PHI_2 | PHI_3 | ... |
| 2 | Y | $\approx$ 35 | T_2 | 1 | PHI_2 | PHI_3 | ... |
| 3 | Y | 65 | T_2 | 1 | PHI_2 | PHI_3 | ... |
| 3 | Parkinson | 65 | Old | 1 | 1 | 0.6 | ... |
| 3 | Migraine | 65 | Middle | 1 | 1 | 0.4 | ... |
The next tuple to be processed is the one above the double line:

<table>
<thead>
<tr>
<th></th>
<th>Y</th>
<th>[54 – 56]</th>
<th>T2</th>
<th>1 PH12</th>
<th>PH13</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Y</td>
<td>[54 – 56]</td>
<td>T2</td>
<td>1 PH12</td>
<td>PH13</td>
</tr>
<tr>
<td>3</td>
<td>Parkinson</td>
<td>65</td>
<td>Old</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>Migraine</td>
<td>65</td>
<td>Middle</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Migraine</td>
<td>35</td>
<td>Middle</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The next tuple to be processed is the one above the double line:

<table>
<thead>
<tr>
<th></th>
<th>Y</th>
<th>[54 – 56]</th>
<th>T2</th>
<th>1 PH12</th>
<th>PH13</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Parkinson</td>
<td>65</td>
<td>Old</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>Migraine</td>
<td>65</td>
<td>Middle</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Migraine</td>
<td>35</td>
<td>Middle</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Migraine</td>
<td>54 – 56</td>
<td>Middle</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Parkinson</td>
<td>54 – 56</td>
<td>Old</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The calculation of the facts the predicate MaySufferFrom results in the following relation:

<table>
<thead>
<tr>
<th></th>
<th>Parkinson</th>
<th>Min(1,1,0.6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Parkinson</td>
<td>Min(1,1,0.4)</td>
</tr>
<tr>
<td>2</td>
<td>Migraine</td>
<td>Min(1,1,1)</td>
</tr>
<tr>
<td>1</td>
<td>Migraine</td>
<td>Min(1,1,0.6)</td>
</tr>
<tr>
<td>1</td>
<td>Parkinson</td>
<td>Min(1,1,0.7)</td>
</tr>
</tbody>
</table>

6. Conclusion and future work

A definition for generalized rule that can represent and handle imprecise information has been given and we have set out the main features of an extended algorithm for deducing with fuzzy data by the application of these generalized rules.

In the future, the negation of a predicate in the head or in the body of a generalized rule will be also discussed. Furthermore, the use of uncertainty in rules as in fuzzy Prolog will be studied as well as the incorporation of other models capabilities (such as FRIL) to a GEFRED database to be used in deductions.

Acknowledgments

This work has been partially supported by the Spanish “Comisión Interministerial de Ciencia y Tecnología” under grant TIC97-0931, the Spanish ”Ministerio de Ciencia y Tecnología” (MCYT) under grant CICYT TIC1999-0558, the European Project FEDER 1FD97-0255-C03-02 and the Spanish ”Ministerio de Ciencia y Tecnología” (MCYT) project TIC2002-00480.

References

25. P. Kullman and S. Sandri. Implementation of an extended possibilistic logic in an
annotated logic theorem prover. In Proceedings of Joint 9th IFSA World Congress
27. D. Liu and D. Li. A new fuzzy inference language f-prolog. Computer Engineering,
28. T. P. Martin and F. Arcelli Fontana, editors. Logic Programming and Soft Comput-
fuzzy relational database. In Springer, editor, Lecture Notes in Artificial Intelligence,
volume 1079, pages 491–500. 1996.
613, 1997.
32. J. M. Medina, M. A. Vila, J. C. Cubero, and O. Pons. Towards the implementation of
a generalized fuzzy relational database model. Fuzzy Sets and Systems, 75:273–289,
1995.
34. R. Ng and V. S. Subrahmanian. Probabilistic logic programming. Information and
35. H. Prade. Lipaki's approach to incomplete information databases restated and
generalized in the setting of zadeh's possibility theory. Information Sciences, 9:27–42,
1984.
36. H. Prade and C. Testemale. Generalizing database relational algebra for the treat-
ment of incomplete/uncertain information and vague queries. Information Sciences,
relational and fuzzy deductive databases. In European Research Consortium for
40. H. E. Vrinten. Linguistic logic programming. In T. P. Martin and F. Arcelli Fontana,
editors, Logic Programming and Soft Computing, pages 91–110. Research Studies
41. P. Votjas, T. Alsinet, and L. Godo. Different models of fuzzy logic programming with
fuzzy unification (towards a revision of fuzzy database model). In Proceedings of Joint
9th IFSA World Congress and 20th NAFIPS International Conference, 2001.
42. E. A. Wong. Statistical approach to incomplete information in database systems. ACM
44. L. A. Zadeh. The concept of a linguistic variable and its application to approximate
45. M. Zemankova and A. Kandel. Fuzzy Relational Databases - A Key to Expert