Fuzzy functional dependencies - An overview and a critical discussion

Patrick Bosc, Didier Dubois, Henri Prade

FUZZY FUNCTIONAL DEPENDENCIES
—AN OVERVIEW AND A CRITICAL DISCUSSION—

Patrick BOSC* – Didier DUBOIS – Henri PRADE****

* I.R.I.S.A./E.N.S.S.A.T., B.P. 447, 22305 Lannion Cedex, France, Email: bosc@enssat.fr

** I.R.I.T., Univ. P. Sabatier, 31062 Toulouse Cedex, France, Email: {dubois, prade}@irit.fr

Abstract

In the context of regular relational databases, functional dependencies have received a lot of attention, since they capture some semantics about the data related to redundancy. Functional dependencies lead to an appropriate design of a database in terms of a set of relations and can make the checking process significantly easier. For about ten years, several proposals to deal with ill-known or weighted data in database management systems have been made and extensions of the relational data model have been proposed accordingly. In this context, the concept of fuzzy functional dependency has emerged, and several definitions have been given. In this paper, a critical overview of different proposals is provided, and the intended semantics and use of fuzzy functional dependencies are particularly discussed.

I. Introduction

An important feature of a database management system (DBMS) is to handle properties that data should fulfil, known as integrity constraints (IC's). These constraints are not invented for technical reasons, but in fact they are reflecting properties of the real world. For instance, if a database contains information about customers, orders and stock, properties such as: "the quantity ordered for a given item is smaller than the quantity of this item appearing in the stock", "to order, a customer must be known (described)", "the delivery address is always the same for a given customer" may hold. One solution to maintain these properties is to rely on ad hoc programs written by database users in order to ensure that no update violates the IC's valid in the base. This approach has several drawbacks, in particular its non declarative aspect. That is why, DBMS's are evolving in order to free users from an explicit programming of the IC's, at least for some types of IC's which could be given in a declarative form.

Among all the possible IC's that can be imagined, a particular type came out very early, along with the relational model of data [7]: functional dependencies (FD's) [2], and to a lesser extent other dependencies (multi-valued [10] and join dependencies [13][16]). In fact, dependencies are IC's with a special status. For a number of reasons, (mainly the risk for inconsistency when updates are performed), redundancy is not convenient in a database and dependencies have emerged since they are capable of capturing some semantics of the data strongly connected with the occurrence of redundancy in a database. Informally, a FD is a property valid on a relation, stating that tuples with the same value on a set A of attributes have the same value on a set B of attributes.

Let us introduce this phenomenon with a database dealing with sales through the relation ORDERS(customer, city, product, price) where the following property (P) holds: "a given customer is located in a unique city". Clearly P is an FD between the attribute customer on the one hand and city on the other hand. If the customer Smith from Boston has ordered the products p1, p2 and p7, the relation will involve the three tuples: <Smith, Boston, p1, $4>, <Smith, Boston, p2, $9>, <Smith, Boston, p7, $6> and it is obvious that the information tied to Smith's city is replicated, thus generating redundancy. This situation illustrates the connection between the existence of an FD and the presence of redundancy in a relation.

Fortunately, it is possible to significantly reduce (and even to totally remove) redundancy from a database if its schema is chosen (or designed) appropriately. In our previous example, let us now assume that we represent our data using two distinct relations: CUST(customer, city) describing the customers and their related city and ORDERS(customer, product, price) dealing with the products ordered by the customers. The information concerning Smith will here be stored as four tuples, one in CUST: <Smith, Boston> and three in ORDERS: <Smith, p1, $4>, <Smith, p2, $9>, <Smith, p7, $6>. Using this representation, no redundancy occurs although the same information is recorded in the database. This example shows the link between redundancy, functional dependencies and database design (through relation splitting).

The removal of redundancy does not change the fact that FD's are valid whatever the schema chosen to model the data. It is then interesting to compare what has to be done to check the validity of FD's depending on the schema. In the previous example, with the first schema involving a single relation, the satisfaction of the property P requires a program checking that, all tuples referring to a given customer, have the same value for the attribute 'city'.

If the second schema is retained, it is worth noticing that the satisfaction of the property P reduces to making sure that the attribute customer remains the key of the CUST relation when this relation is updated. This verification is very easy to implement (since any file system supporting a DBMS provides the uniqueness of declared keys) and, above all, this does not require any explicit program to be written.

Since the beginning of the eighties, several groups of researchers have been working on the application, to database management, of methods based on fuzzy sets and possibility theory for the treatment of imprecision and uncertainty and the handling of properties whose satisfaction is a matter of degree. This objective is very wide and covers several issues such as the extension of the data models to support ill-known data [5][14][19][22], the expression of flexible queries directed to regular or fuzzy databases [18][4][14] and the description of soft constraints bearing on the data. This last direction has mainly been concerned with the extension of functional dependencies (FD's) to fuzzy functional dependencies (FFD's) for which different propositions have been made [15][6][8]. In each proposal, a particular definition for extending the concept of FD is suggested and its justification often seems to reside more on the validity of "good" mathematical properties (analogous to those valid for FD's) than on their intended semantics and their intended use. In particular, it is not made clear what FFD's bring from the point of view of redundancy (and schema design). In this paper we rather consider FFD's as IC's (redundancy and database design issues are left out) and propose a general framework for extending the notion of FD using a typology of fuzzy rules.

The structure of the paper is the following. In Section 2, the main results concerning classical functional dependencies are recalled. Section 3 provides the necessary background on fuzzy databases for discussing FFD's and their intended meanings. Section 4 reviews previous proposals for FFD's and discusses them with respect to regular databases or databases with weighted tuples. Some considerations regarding database design and relation splitting will be pointed out. Section 5 discusses FD's in databases where tuples include fuzzily known values. Section 6 distinguishes two kinds of FFD's and suggests their possible uses for extrapolating values.

II. Principal Results Concerning Functional Dependencies

Functional dependencies have been defined in the context of data modelled as relations. A relation R over a set of domains $D_1, \ldots, D_n$ is any subset of the Cartesian product of these domains. In the following, U will represent the universe over which a relation R is defined. The functional dependency $X \rightarrow Y$ holds over R(U) iff:

$$\forall t1, t2 \in R, t1.X = t2.X \Rightarrow t1.Y = t2.Y \quad (1)$$

with $X, Y \subseteq U$ and where t.X denotes the restriction of the tuple t to the attributes belonging to X.

FD's fulfil a number of properties such as reflexivity, augmentation (if $X \rightarrow Y$ then $X \cup Z \rightarrow Y$), and transitivity (which are formalized by Armstrong's axioms). It is possible to focus on a family known as elementary functional dependencies (EFD's) which are of special interest since they represent "atomic" dependencies and they allow for the removal of trivial FD's (due to the property of reflexivity: $\forall X, Y \subseteq U, X \subseteq Y \Rightarrow Y \rightarrow X$). $X \rightarrow Y$ is an EFD iff: i) $X \rightarrow Y$ is an FD, ii) Y is a single attribute, iii) Y is not included in X and iv) there is no proper subset X' of X such that $X' \rightarrow Y$. The notion of key of a relation is also connected to that of FD. $C \subseteq U$ is a key of R(U) iff: i) $C \rightarrow U$ and ii) there is no proper subset C' of C such that $C' \rightarrow U$.

Given these definitions, let us consider a relation R where the EFD: $X \rightarrow Y$ holds. If X is a key of R, there is no redundancy in R (with respect to this FD) and the satisfaction of the constraint represented by this FD reduces to that of the uniqueness of the values of the set X of attributes. If X is not a key, the FD causes some redundancy in R and the satisfaction of the FD will require a program to check it. The work called "normalization" in the relational database theory is exploiting this idea, i.e., one tries to have all EFD's captured by keys of relations since these FD's do not generate redundancy. Moreover, in this case, FD checking reduces to maintaining the uniqueness of key values in the relations.

Another point worthy of attention concerns the expression of FD's in a schema. Let us assume a universe where $X \rightarrow Y$ holds and consider a schema where this FD is lost, i.e., no relation possesses the attributes X and Y together. In this case, the maintenance of the property $X \rightarrow Y$ will be difficult and will require a specific program to be written to this aim. Let us illustrate this situation with the schema involving the two relations: R1(C,V) and R2(C,R) assuming that the FD's: $C \rightarrow V$ and $\{V,R\} \rightarrow C$ are valid on the universe {C,V,R}. With this representation, the first FD is represented in R1 but the second is lost.

Normal forms of relations have been defined to constrain the type of FD's that are allowed in the relations of a schema. Third normal form (3NF) states that the only EFD's $X \rightarrow Y$ allowed in a relation are such that X is a key or Y belongs to a key, whereas Boyce-Codd-Kent normal form (BCNF) imposes that the only EFD's present in a relation have a key in the left-hand side. Clearly, BCNF relations correspond to our objective in terms of redundancy, but it has been shown that it is not always possible to model a universe as a set of BCNF relations where all the initial FD's are preserved, while it is possible for 3NF relations. This result tells us that, in general, it

is necessary to write programs to check an FD X → Y either because X is not a key of a relation (a 3NF schema has been chosen) or because this FD has been lost in the adopted representation (a BCNF schema has been chosen).

The transformation of a schema is based on the following decomposition theorem: the relation R(X,Y,Z) with the FD X → Y can be replaced by its two projections R[X,Y] and R[X,Z]. This theorem guarantees that the data represented in the two schemas are identical, because R can be reconstructed from its projections using a natural join. This is the notion known as lossless join property of relations.

Example. Let us consider the universe {C,V,R} with two EFD's: C → V and {V,R} → C. The representation of this universe as a single relation S(V,C,R) leads to a 3NF schema but the FD: C → V is such that C is not a key of S (which has two keys {V,R} and {C,R}) and some redundancy can take place. This schema may be transformed according to the theorem into a schema involving two relations: S1 = S[C,V] and S2 = S[C,R]. In this lattter schema, S1 and S2 are clearly BCNF, but the FD {V,R} → C is lost.

From a computational point of view, algorithms have been designed in order to generate 3NF and BCNF schemas [3]. They need to compute keys, which in turn require the knowledge of a minimal cover of the valid FD's. A minimal cover can be found using Armstrong's axioms [1] which proved to be a complete and valid set of rules, i.e., any valid FD is found and any FD which is found does hold. We will see later that some extensions of FD's into fuzzy FD's are interested in the preservation of such a deduction system over dependencies (FFD's).

III. Fuzzy Databases and Fuzzy Functional Dependencies

In the following we only consider two kinds of fuzzy relational databases. For the sake of brevity, we do not discuss Buckles and Petry's approach [5] where fuzzy similarity relations are attached to attribute domains to model interchangeability between values. Namely we consider databases where tuples are "ordinary" tuples associated with a weight (usually belonging to [0,1]) and those where the tuples may include any (fuzzy) subset of values of the attribute domains rather than precise and unique attribute values as in ordinary tuples. For instance, consider a relation made of the attributes Name, Age, Salary (Peter, 25, 20000; 0.8) is a weighted tuple, and (Paul, YOUNG, [20000,22000]) is a tuple with a fuzzy value (YOUNG) and an imprecise value ([20000, 22000]). In order to define meaningful operations, it is important to explain the intended meanings of the weight in the first case, and of the subsets in the second case. There are mainly two types of interpretations for the fuzzy weight: it is either a global confidence level in the information stored in the tuple, or it may be the degree to which the tuple belongs to the relation which is then supposed to have a fuzzy meaning. In the above example this second interpretation seems difficult; however consider the relation LIKES with attributes Person's name, Movie's name, then the weight associated with a tuple may in this case either mean the confidence in the information as said above (then LIKE is interpreted in a binary way), or estimate the extent to which the person likes the movie (then LIKE is interpreted as a gradual property). Note that it is usually assumed that identical tuples with different weights are not allowed whatever the interpretation of the weight (viewing the weight as a special attribute, there is a FD <Attributes of the tuple> → <Weight>!). A collection of weighted tuples over a given set of attributes defines a fuzzy relation (i.e., a fuzzy subset of a Cartesian product).

In the case of tuples with imprecise or fuzzy components, the subsets, fuzzy or not, are restricting the more or less possible values of a *single*-valued attribute. In other words, the membership function of the (fuzzy) subset is interpreted as a possibility distribution. In the above example, $\mu_{YOUNG}(a)$ estimates the degree of possibility that age(Paul) = a. This interpretation of subsets completely departs from the case of a multiple-valued attribute like Spoken languages, as in the example (Paul{English,French}) where the tuple would mean that Paul speaks both English *and* French (while in case of a single-valued attribute like "Native language", {English,French} would mean "either English or French". It is also worth noticing that a tuple of (fuzzy) subsets can be viewed to some extent as equivalent to a collection of a special kind of (weighted) tuples. This is particularly true in the case of multiple-valued attributes, for instance in the relation Spoken languages (Paul,{English,French}) is equivalent to the two tuples {(Paul,English),(Paul,French)}. In the case of imprecisely known single-valued attributes, the same remark applies except that the tuples will be mutually exclusive. In the fuzzy case weighted tuples are obtained, whose semantics differs for multiple-valued attributes and single-valued attributes; in the latter case, the weight is a possibility degree. Moreover a relation made of tuples having fuzzy components modelling imprecise values will be turned into a more complex relational structure made of a collection of disjunctive subsets of weighted tuples. This remark indicates that tuples with fuzzy components is a compact representation of an otherwise complex relational structure. Moreover it is a faithful representation provided that the attributes be non-interactive.

The idea of representing not only regular integrity constraints but also soft ones in order to take advantage of all the available information (even if it

is incomplete or imprecise) is clearly independent of the notion(s) of fuzzy datbases. In other words, it may be fruitful to distinguish between the use of fuzzy functional dependencies in the case of a regular database, and the problems raised by the application of regular functional dependencies to a fuzzy database. Obviously, once these two questions are clarified, we can consider the general case of a fuzzy functional dependency applying to a fuzzy database.

IV. Previous Approaches to Fuzzy Functional Dependencies

We will briefly present a few proposals for extending FD's into FFD's. Basically, the idea is to start with the definition (1) of an FD and to transform it in order to take into account situations where the strict equality does not apply. Consequently, resemblance relations (reflexive and symmetric) over data domains are introduced ($RES_X(t1.X, t2.X)$ will denote the resemblance degree between t1.X and t2.X). Resemblance between tuples are computed by a conjunctive aggregation of the resemblances between attribute values; usually the minimum operation is used, although we might think of other operations.

Raju and Majumdar [15] have proposed the following extension of FD, X ~> Y is valid on R(U) $\Leftrightarrow \forall t1, t2 \in R$,

$$RES_X(t1.X, t2.X) \Rightarrow_{RG} RES_Y(t1.Y, t2.Y) \quad (2)$$

where $\Rightarrow_{RG}$ stands for Rescher-Gaines' implication ($\alpha \Rightarrow_{RG} \beta = 1$ if $\alpha \leq \beta$, 0 otherwise). It applies both to classical databases and to fuzzy relations made of weighted tuples. Note that FFD's obeying (2) are stronger than (1) (i.e., FD's) iff restricted relations are used (i.e. relations such that $RES_D(a,a') = 1 \Leftrightarrow a = a'$) since (2) also reads $RES_X(t1.X, t2.X) \leq RES_Y(t1.Y, t2.Y)$; otherwise (2) is not comparable with (1). Note that except for the 1-cut (in case of a restricted resemblance), the α-cut of a FFD, i.e., $RES_X(t1.X, t2.X) \geq \alpha \Rightarrow RES_Y(t1.Y, t2.Y) \geq \alpha$ is not a FD, but a more general IC. It can be shown that [15][12]: i) Armstrong's axioms apply to this type of FFD and define a valid and complete deduction system for FFD's, ii) if *restricted* resemblance relations are used, then the following decomposition theorem holds:

R(X,Y,Z) with the restricted FFD X ~> Y can be replaced by the projections R[X,Y] and R[X,Z]. (3)

G. Chen et al. [6] have considered a slightly different definition, also in the scope of fuzzy relations:

$X \sim>_\lambda Y$ ($\lambda \in]0,1]$) is valid on R(U)

$\Leftrightarrow \forall t1, t2 \in R$, $t1.X = t2.X \Rightarrow t1.Y = t2.Y$ ***and***

$$[RES_X(t1.X, t2.X) \Rightarrow_G RES_Y(t1.Y, t2.Y)] \geq \lambda \quad (4)$$

where $\Rightarrow_G$ denotes Gödel's implication ($\alpha \Rightarrow_G \beta = 1$ if $\alpha \leq \beta$, β otherwise). This proposal strenghens definition (1) in adding a constraint when values are no longer equal, but only similar according to resemblance relations. Chen et al. have established the following results: i) it is possible to build an extended set of Armstrong's axioms defining a valid and complete set of deduction rules for this kind of FFD's, ii) the decomposition theorem (3) holds without any constraint over resemblance relations and iii) fuzzy normal forms have been designed, whose definition is the usual one where the term FD has been replaced by FFD. It can be noticed that this definition allows some discontinuity since $t1.X = t2.X \Rightarrow t1.Y = t2.Y$, but if t1.X and t2.X are very close, it is sufficient that t1.Y and t2.Y resemble each other at degree λ, which does not mean that they are very close. Note that in these two proposals the new property expressed by the FFD remains a Boolean one and is fully satisfied or unsatisfied. This is also true with Cubero et al.'s proposal [8] where $X \sim>_{(\alpha,\beta)} Y$ is defined by $RES_X(t1.X, t2.X) \geq \alpha \Rightarrow RES_Y(t1.X, t2.X) \geq \beta$. Kiss [11] also works with weighted tuples, using strict equality, but takes into account the weights by defining the degree to which the FFD holds, as

$$1 - \sup_{t1,t2:t1.X=t2.X \text{ and } t1.Y \neq t2.Y} \min(\text{weight}(t1), \text{weight}(t2)). \quad (5)$$

Let us also mention that Shenoi et al. [17] have proposed a definition of FFD's in Buckles and Petry's framework, in the spirit of [15].

In fact, in each of these extensions, a specific choice is made, but this choice is not generally governed by semantic considerations, but rather by the fact that a parallel can be made between properties valid for FD's and for the considered FFD's. There is no reason for choosing one and not the other as far as these definitions correspond to actual constraints bearing on the database. The motivation for introducing FFD's such as (2), (4) or (5) seems to be the preservation of properties of classical FD's rather than an attempt to model some real-world constraint relating the components in tuples. We will have the same difficulty to explain the meaning for definition (4), especially with the appearance of the threshold λ. We believe that it is important to have a clear meaning of these properties if we want people to express them in the context of a given real world modelling. Let us for instance consider the FD Volume, Matter → Weight. In such a case, it is clear that not only two items in the same matter with the same volume have the same weight, but also that if they are in the same matter, their weights should be close when their volumes are close. In such an example, it is certainly possible to define the fuzzy resemblance relations in such a way that the FFD agrees with the physical reality.

Another question concerns the interest of the decomposition of a relation where the FFD X ~> Y holds. When the FD X → Y holds on a relation, the decomposition is envisaged as a way to remove

redundancy and to facilitate the checking process of the FD. With a FFD, this objective is no longer achievable since, even in the schema issued from the decomposition, the satisfaction of the FFD will remain a matter of resemblance, and thus, will require pairwise comparisons of tuples. The conclusion here is that FFD's, as defined above, are interesting in terms of data properties, i.e., integrity constraints, but, unlike FD's, they do not seem to be connected with the schema design of a database (except if their definition is stronger than a regular FD). Indeed, the decomposition theorem (3) holds in the cases above when the definitions are stronger than classical FD's. It is easy to show that a fuzzy relation R(X,Y,Z) with the classical FD $X \rightarrow Y$ is equivalent to R[X,Y] and R[X,Z]. Then, any property on R which implies this FD will make R decomposable.

V. (F)FDs in Fuzzy Databases

Let us consider the case of a regular FD in a fuzzy database where tuples include subsets of attribute values interpreted as possibility distributions. Observe first that if two quantities x and y are known to be equal, the possibility distributions π_x and π_y which restrict their possible values in case of incomplete information should be equal, i.e., $x = y \Rightarrow \pi_x = \pi_y$, but the converse is false. Knowing that $\pi_x = \pi_y$ it is only *possible* that x = y (when π_x and π_y are not membership functions of singletons). Thus if imprecise fuzzy attribute values are allowed, it is necessary to understand the meaning of the expression $RES_D(A1,A2)$ where $\pi_{t.X1} = \mu_{A1}$ and $\pi_{t.X2} = \mu_{A2}$. In [15], it is suggested to define the resemblance between A1 and A2, as: $I_{A1 \approx A2} = \max(\text{card}(A1 \cap A2) / \text{card}(A1), \text{card}(A1 \cap A2) / \text{card}(A2))$. Unfortunately, this way of doing is adequate to compare two fuzzy sets, but not for possibility distributions representing imprecise values. An illustration is when A1 and A2 are completely unknown (any value of the universe is totally possible). In that case $\text{card}(A1 \cap A2) = \text{card}(A1) = \text{card}(A2)$, which would mean that these two values are fully similar. The only true information is that the resemblance is totally possible (but not certain). In fact we have to distinguish between the possibility of resemblance, defined by

$$\prod(t1.X \approx_X t2.X) = \sup_{u \in D_X, u' \in D_X} \min(\pi_{t1.X}(u), \pi_{t2.X}(u'), RES_X(u,u'))$$

and the certainty of resemblance by

$$N(t1.X \approx_X t2.X) = 1 - \prod(t1.X(\text{not} \approx_X)t2.X)$$

where not $\approx_X$ is defined by $1 - RES_X$. Then the monotonicity of $\prod$ and N applied to a FFD obeying (2) leads to the constraints (not acknowledged in [8])

$$\prod(t1.X \approx_X t2.X) \leq \prod(t1.Y \approx_Y t2.Y)$$

$$N(t1.X \approx_X t2.X) \leq N(t1.Y \approx_Y t2.Y)$$

Now, consider the example

Name	Age	Salary
Paul	YOUNG	MEDIUM
Peter	YOUNG	RATHER-SMALL

This database may be perfectly in agreement with the *regular* FD Age → Salary (assumed to hold) provided that there is a value completely compatible with both MEDIUM and RATHER-SMALL, i.e., the consistency of MEDIUM and RATHER-SMALL is equal to 1. Then the possibility is 1 that there exists an instantiation compatible with the fuzzy information and which does not violate the FD. It shows that although t1.Age = t2.Age ⇒ t1.Salary = t2.Salary holds in this example, the FD does not apply to fuzzy values (the equality of fuzzy values for Age does not entail equality for Salary). Furthermore the relation is no longer decomposable except if we accept to lose information by keeping *only* the tuple (YOUNG, MEDIUM ∪ RATHER-SMALL) in the relation Age × Salary. In fact, if we decompose, the situation is still worse, since this tuple would mean that for *any* age restricted by YOUNG the salary is restricted by the other fuzzy set in the tuple. It disagrees with the database where this is only known for two particular Age values: Peter's age and Paul's age. This means that strictly speaking the key Age of the relation (according to the FD) is not allowed to have fuzzy values; it is still more striking if we consider the above example where YOUNG is replaced by UNKNOWN.

VI. Two Kinds of Fuzzy Dependencies

In the scope of modeling real-world constraints that relate the components of tuples, there exists several kinds of fuzzy rules, especially certainty rules (of the form "the more X is A, the more certain <something>"), and gradual rules (of the form "the more X is A, the more Y is B") [9]. This leads to imagine two ways of relating resemblance degrees in generalized functional dependencies. If R and S model resemblance relations, a first kind of dependency expresses that the more similar (in the sense of R) t1.X and t2.X are, i.e., the closer to 1 $\mu_R(t1.X, t2.X)$, the more certain the similarity (as described by S) of t1.Y and t2.Y is. It is expressed by the following constraint on the conditional possibility distribution $\pi_{(t1.Y,t2.Y)|(t1.X,t2.X)}$ (denoted by $\pi_{y,y'|x,x'}$ for simplicity) representing the dependency:

$$\pi_{y,y'|x,x'}(v,v',u,u') \leq \max(\mu_S(v,v'), 1 - \mu_R(u,u')) \quad (6)$$

where u, u', v, v' respectively denote the current value of the variables x = t1.X, x' = t2.X, y = t1.Y and y' = t2.Y. Indeed when $\mu_R(u,u')$ is close to 1, the values v and v' which are not similar in the sense of S should have a low degree of possibility $\pi_{y,y'|x,x'}(v,v',u,u')$. In the particular case where R is a strict equality

relation, we see that (6) expresses that if t1.X = t2.X then t1.Y should resemble t2.Y in the sense of S.

A second kind of dependency is gradual and relates the resemblance of t1.X and t2.X, to the resemblance of t1.Y and t2.Y in the following way

$$\mu_S(v,v') \geq \min(\pi_{y,y'|x,x'}(v,v',u,u'), \mu_R(u,u')) \quad (7)$$

i.e., the closer to 1 $\mu_R(u,u')$ and the more possible the values v and v', the closer to 1 $\mu_S(v,v')$ should be. In the particular case where $\pi_{y,y'|x,x'}$ is a *crisp* possibility distribution, we recover the definition (2) of a fuzzy functional dependency, namely

$$\mu_R(t1.X, t2.X) \leq \mu_S(t1.Y, t2.Y).$$

We examine now what can be deduced on t2.Y with these dependencies when some (fuzzy) information are available on t1.X, t2.X and t1.Y.

• *Certainty rule dependency*: Let us suppose that we know that $\pi_x = \mu_A$; $\pi_{x'} = \mu_{A'}$ and $\pi_y = \mu_B$ which express that the possible values of x, x' and y are restricted by A, A' and B respectively. Then applying Zadeh combination/projection principle of possibility theory, i.e.

$$\pi_{y'}(v') = \sup_{u,u',v} \min(\pi_x(u),\pi_{x'}(u'),\pi_y(v), \pi_{y,y'|x,x'}(v,v',u,u')) \quad (8)$$

we get,with A, A', B normalized, and using (6)

$$\pi_{y'}(v') = \max(\mu_{B \circ S}(v'), 1 - N(R; A \times A')) \quad (9)$$

where N(R; A × A') is the necessity measure of the (fuzzy) event "(x,x') satisfies R" given the information that (x,x') is restricted by the Cartesian product A × A'. (9) expresses that we are certain at the degree N(R; A × A') that y' is restricted by B ∘ S (i.e., the set of values restricted by B or similar in the sense of S to an element in B).

• *Gradual rule dependency:* if we know that $\pi_x = \mu_A$; $\pi_{x'} = \mu_{A'}$; $\pi_y = \mu_B$, we get from (7) and (8)

$$\pi_{y'}(v') = \sup_{u,u',v} \min(\mu_A(u), \mu_{A'}(u), \mu_B(v), \mu_R(u,u') \rightarrow \mu_S(v,v'))$$

where → is Gödel implication.

Thus fuzzy functional dependencies might be useful for tasks such as integrity checking, or the extrapolation of missing values, as well as for analogical reasoning and cooperative answering. Dependencies modeled by gradual rules look particularly adapted to interpolation purposes.

References

[1] W.W. Armstrong, Dependency structures of data base relationships. *Inf. Proces.*, 1974, 580-583.

[2] P.A. Bernstein, J.R. Swenson, D.C. Tsichritzis, A unified approach to functional dependencies and relations. *Proc. ACM SIGMOD Conf.*, San José, 1975, 237-245.

[3] P.A. Bernstein, Synthesizing third normal form relations from functional dependencies. *ACM Trans. on Database Systems*, 1(4), 1976, 277-298.

[4] P. Bosc, O. Pivert, Some approaches for relational databases flexible querying. *J. of Intell. Information Systems*, 1, 1992, 323-354.

[5] B.P. Buckles, F.E. Petry, A fuzzy representation of data for relational databases. *Fuzzy Sets & Systems*, 5, 1982, 213-226.

[6] G.Q. Chen, J. Vandenbulcke, A step towards the theory of fuzzy relational database design. *Proc. Inter. Fuzzy Systems Association (IFSA'91) Congress*, Brussels, Belgium, Vol.: Computer, Management & Systems Science (R. Lowen, M. Roubens, eds.), 1991, 44-47.

[7] E.F. Codd, A relational model of data for large shared data banks. *Communications of the ACM*, 13(6), 1970, 377-387.

[8] J.C. Cubero, J.M. Medina, M.A. Vila, Influence of granularity level in fuzzy functional dependencies. In: *Symbolic and Quantitative Approaches to Reasoning and Uncertainty (Proc. ECSQARU'93)*, LNCS, Vol. 747, Springer Verlag, 1993, 73-78.

[9] D. Dubois, H. Prade, The semantics of fuzzy "if… then…" rules. In: Fuzzy Approach to Reasoning and Decision Making (V. Novák, J. Ramík, M. Cerny, Nekola J., eds.), Kluwer, 1992, 3-16.

[10] R. Fagin, Multivalued dependencies and a new normal form for relational databases. *ACM Trans. on Database Systems*, 2(3), 1977, 262-278.

[11] A. Kiss, λ-decomposition of fuzzy relational databases. *Annales Univ. Sci. Budapest., Sect. Comp.*, 12, 1991, 133-142.

[12] Liu, The reduction of the fuzzy data domain and fuzzy consistent join. *Fuzzy Sets Syst.*, 50, 89-96.

[13] J-M. Nicolas, Mutual dependencies and some results on undecomposable relations. *Proc. VLDB Conf.*, Berlin, 1978, 360-367.

[14] H. Prade, C. Testemale, Generalizing database relational algebra for the treatment of incomplete/ uncertain information and vague queries. *Information Sciences*, 34, 1984, 115-143.

[15] K.V.S.V.N. Raju, A.K. Majumdar, Fuzzy functional dependencies and lossless join decomposition of fuzzy relational database systems. *ACM Trans. on Database Systems*, 13(2), 1988, 129-166.

[16] J. Rissanen, *Theory of Relations for Databases — A Tutorial Survey*. LNCS, Vol. 64, Springer Verlag, Berlin, 1978, 537-551.

[17] S. Shenoi, A. Melton, L.T. Fan, Functional dependencies and normal norms in the fuzzy relational database model. *Infor. Sci.*, 60, 1992, 1-28.

[18] V. Tahani, A conceptual framework for fuzzy query processing: A step toward very intelligent database systems. *Information Processing Management*, 13, 1977, 289-303.

[19] M. Umano, FREEDOM-0: A fuzzy database system. In: *Fuzzy Information and Decision Processes* (M.M. Gupta, E. Sanchez, eds.), North-Holland, Amsterdam, 1982, 339-347.

[20] L.A. Zadeh, Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets & Systems*, 1, 1978, 3-28

[21] C. Zaniolo, A new normal form for the design of relational database schemata. *ACM Trans. on Database Systems,* 7(3), 1982, 489-492.

[22] M. Zemankova, A. Kandel, *Fuzzy Relation Data Bases — A Key to Expert Systems*. Verlag TÜV Rheinland, Köln, 1984.