NIFS 13 (2007), 1, 20-33

Representation of Null Values with the Aid H-IFS

Panagiotis Chountas

HSCS – University of Westminster, London, HA1 3TP, UK
chountp@wmin.ac.uk

Abstract. Here, we present a way to replace unknown values using background knowledge of data that is often available arising from a concept hierarchy, as integrity constraints, from database integration, or from knowledge possessed by domain experts. We present and examine the case of H-IFS to represent support contained in subsets of the domain as a candidate for replacing unknown values mostly referred in the literature as NULL values.

Keywords: Null values, Intuitionistic Fuzzy Sets, Knowledge Representation

1 Introduction

Background knowledge of data is often available, arising from a concept hierarchy, as integrity constraints, from database integration, or from knowledge possessed by domain experts. Frequently integrated DBMSs contain incomplete data which we may represent by using H-IFS to declare support contained in subsets of the domain. These subsets may be represented in the database as partial values, which are derived from background knowledge using conceptual modelling to re-engineer the integrated DBMS. For example, we may know that the value of the attribute JOB-DESCRIPTION is unknown for the tuple relating to employee Natalie but also know from the attribute salary that Natalie receives an estimated salary in the range of €25K ~Salary$_{25K}$. A logic program, using a declarative language can then derive the result that Natalie is a "Junior-Staff", which we input to the attribute JOB-DESCRIPTION of tuple Natalie in the re-engineered database. In such a manner we may use the knowledge base to replace much of the unknown in the integrated database environment.

Generalised relations have been proposed to provide ways of storing and retrieving data. Data may be imprecise, hence we are not certain about the specific value of an attribute but only that it takes a value which is a member of a set of possible values. An extended relational model for assigning data to sets has been proposed by [1]. This approach may be used either to answer queries for decision making or for the extraction of patterns and knowledge discovery from relational databases. It is therefore important that appropriate functionality is provided for database systems to handle such information.

A model, which is based on partial values [2], has been proposed to handle imprecise data. Partial values may be thought of as a generalisation of null values where, rather than not knowing anything about a particular attribute value, we may identify the attribute as a set of possible values. A partial value is therefore a set such that exactly one of the values in the set is the true value.

We review the different types of NULL values and then we focus is on providing an integrated DBMS environment that will enable us to:

- reconcile unknown information with the aid of background knowledge. We examine the appropriateness of the H-IFS as a form of Background knowledge for replacing unknown attribute values
- utilise constraints as part of the integrated DBMS metadata to improve query execution
- query imprecise information a part of integrated DBMS environment that may entail more than one sources of information

2 Review of NULL Values

A null value represents an *unknown* attribute value, is a value that is known to exist, but the actual value is unknown. The unknown value is assumed to be a valid attribute value, that is, some value in the domain of that attribute. This is a very common kind of ignorant information. For example, in an employee database, while everyone must have a surname, Alex's surname may be recorded as unknown. The unknown value indicates that Alex has a name, but we do not know her name. An unknown value has various names in the literature including *unknown null* [3], *missing null* [4], and *existential null* [5].

The meaning of a fact, *F*, with an unknown attribute value over an attribute domain of cardinality *N* is a multiset with *N* members; each member is a set containing an *F* instance with the unknown value being replaced by a different value from the attribute domain. For example, assume $f = \{\text{IBM }(\perp)\}$ where ($\perp$ represents an unknown value over a domain {20, 22} with respect to IBM's share-price), then the meaning of *f* is

$$f = \{\{\text{IBM }(20)\}, \{\text{IBM }(22)\}\}$$

This corresponds to the notion that a fact with an unknown value is incomplete with respect to a fact where that unknown value is no longer unknown, but is now known to be a specific value (i.e. $f_1 = \{\{\text{IBM }(20)\}\}$).

Another generalization of an unknown fact is a *disjunctive* fact [6], so known as *indefinite* information [7]. A disjunction is a logical *or* applied to fact instances. Let *F* be an inclusive disjunctive fact with *N* disjuncts. The meaning of *F* is given by a multiset with *N* members; each member is a set containing one disjunct. For example, the share price of IBM may be £20 or £22. (i.e." IBM (20), IBM (22)"). The disjunction is *exclusive* [8] or *inclusive* [9]. If it is an exclusive disjunction, one and only one disjunct is true. The meaning of an exclusive disjunctive fact is the same as that of an imprecise value. Let $f = \{\{\text{IBM }(20)\}, \{\text{IBM }(22)\}\}$ be an exclusive disjunctive then the meaning of *f* is $f = \{\text{IBM }(20)\} \vee \{\text{IBM }(22)\}$.

The meaning of an inclusive disjunctive fact is somewhat different than that of its exclusive complement, at least one alternative may be true. Let *F* be an inclusive disjunctive fact with *N* disjuncts. The meaning of *F* is given by a multiset with 2^N-1 members; each member is a unique subset of disjuncts. For example, assume, the inclusive disjunct $f = \{\text{IBM }(20)\ \{\text{IBM }(22)\}$ then the meaning of *f*, is $f = \{\{\text{IBM }(20)\}, \{\text{IBM }(22)\}, \{\text{IBM }(20), \text{IBM }(22)\}\}$, excluding the fact, $\{\{\text{IBM }(\perp)\}$. The empty ($\perp$) attribute represents the situation where a fact instance exists, but does not have a particular attribute-label value.

A *maybe value* is an attribute-label value which might or might not exist [10]. If it does exist, the value is known. A *maybe tuple or fact-instance* is similar to a maybe value, but the entire tuple might not be part of the relation. Maybe tuples are produced when one disjunct of an inclusive disjunctive fact-instance is found to be true.

A combination of inclusive disjunctive fact instance and a maybe fact instance can determine the semantics of *open information* or nulls [11]. The denotation of an open null is exact to inclusive disjunctive information with the addition of the empty set as a possible value. That is the attribute-lable value may not exist, could be exactly one value, or could be many values. For example, in the shares database an open value could be used to present IBM share prices. This value means that IBM share price possibly had a past record, (this could be the first appearance in the market); IBM share price may be one or many. The open value covers all this possibilities. A generalization of open information is *possible* information [12] (this differs from our use of the term "possible"). Possible information is an attribute value whose existence is undetermined, but if it does exist, it could be multiple values from a *subset* of the attribute domain.

A *no information* value is a combination of an open value and an unknown value [13]. The no information value restricts an open value to resemble an unknown value. A, no information value might not exist, but if it does, then it is a single value, which is unknown, rather than possibly many values. The meaning of a no information value is similar to that of an *unknown* value with the inclusion of the addition of the empty set as a possible value.

Unknown, partially known, open, no information, and maybe null values are different interpretations of a null value. There are other null value interpretations, but none of these is a kind of well cognisant information.

An *inapplicable* or *does not exist* null is a very common null value. An inapplicable null, appearing as an attribute value, means that an attribute does not have a value [14]. An inapplicable value neither contains nor represents any ignorance; it is known that the attribute value does not exist. Inapplicable values indicate that the schema (usually for reasons of efficiency or clarity) does not adequately model the data. The relation containing the inapplicable value can always be decomposed into an equivalent set of relations that do not contain it. Hence the presence of inapplicable values indicates inadequacies in the schema, but does not imply that information is being incompletely encoded.

Open nulls is the main representative of the possible-unweighted–unrestricted branch. Universal nulls may also be classified under this branch assuming the OWA semantics. Inclusive disjunctive information, possible information and maybe tuples or values are indicative representatives of the possible-unweighted-restricted school.

In [15] five different types of nulls are suggested. The labels and semantics of them are defined as follows. Let V be a function, which takes a label and returns a set of possible values that the label may have.

Label (x)	**V(x)**
Ex-mar	D
Ma-mar	$D \cup \{\perp\}$
Pl-mar	$\{\perp\}$
Par-mar (V_s)	V_s
Pm-mar (V_s)	$V_s \cup \{\perp\}$

Fig. 1. Types of NULL and their semantics

Intuitively, V (Ex-mar) = D says that the actual value of an existential marker can be any member of the domain D. Likewise, V (Ma-mar) = $D \cup \{\perp\}$ says that the actual value of a maybe marker can be either any member of D, or the symbol ⊥, denoting a non-existent value. Similarly, V (Par-mar (V s)) = V_s says that the actual value of a partial null marker of the form pa mar (V_s) lies in the set V_s, a subset of the domain D.

An important issue is the use of ⊥, which denotes that an attribute is inapplicable. However such an interpretation of the unknown information, is not consistent with the principles of conceptual modelling. Assuming the sample fact spouse, the individual, Tony, is a bachelor and hence, the wife field is inapplicable to him, ⊥. Conceptually the issue can be resolved with the use of the subtypes (e.g. married, unmarried) as part of the entity class Person. A subtype is introduced only when there is at least one role recorded for that subtype. The conceptual treatment of null will permit us to reduce the table in Fig.1 using only two types of null markers.

Label (x)	V(x)
V-mar (V)	{V}
P-mar (V_s)	{V_s}
Π-mar (D-V_s)	{D –V_s}

Fig. 2. The reduced set of NULL values

In the general case the algebraic issue under the use of subtypes is whether the population of the subtypes in relationship to the super type is:

- Total and Disjoint: Populations are mutually exclusive and collectively exhaustive.
- Non-Total and Disjoint: Populations are mutually exclusive but not exhaustive.
- Total and Overlapping: Common members between subtypes and collectively exhaustive, in relationship to super type.
- Non-Total and Overlapping: Common members between subtypes and not collectively exhaustive, in relationship to super type.

Conclusively it can be said that a null value is often semantically overloaded to mean either an unknown value or an inapplicable. For an extensive coverage of the issues related to the semantic overloading of null values somebody may further refer to [16].

3 NULL Values & Background Knowledge in DBMS

In a generalised relational database we consider an attribute A and a tuple t_i of a relation R in which n attribute value $t_i[A]$ may be a partial value. A partial value is formally defined as follows.

Definition 3.1 A partial value is determined by a set of possible attribute values of tuple t of attribute A of which one and only one is the true value. We denote a partial value by $P = [a_1,...,a_n]$ corresponding to a set of P possible values $\{a_1,... , a_n\}$ of the same domain, in which exactly one of these values is the true value of . Here, P is the cardinality of $\{a_1,... , a_n\}$ is a subset of the domain set $\{a_0,... , a_{n+1}\}$ of attribute A of relation R, and $P \leq n+1$.

Queries may require operations to be performed on partial values; this can result in a query being answered by means of bags, where the tuples have partial attribute values [17].

Example 3.1 We consider the attribute JOB_DESCRIPTION that has possible values 'Research Associate', 'Teaching Associate', ‘Programmer’, 'Junior Staff', and 'Senior Staff'. Then {'Junior Staff', 'Senior Staff'}, is an example of a partial value in terms of classical logic.

In this case we know only that the individual is a staff but not whether he or she is a junior staff or a senior staff.

Definition 3.2 An Intuitionistic Fuzzy partial value relation R, is a relation based on partial values for domains $D_1, D_2, \ldots, D_n$ of attributes $A_1, A_2, \ldots, A_n$ where $R \subseteq P_1 \times P_2 \times \times P_n$ and P_i is the set of all the partial values on power set of domain D_i. A pruned value of attribute A_i of the relation R corresponds then to a **H-IFS** which is a subset of the domain D_i. An example of a partial value relation is presented in Table 1 below:

Table 1. Generalised Relation Staff with partial values in the form of hierarchical IFS

Name	JOB_DESCRIPTION	SALARY
Natalie	{Research Associate/<1.0,0>}	$\{\sim Salary_{25K}\}$
Anna	{Programmer/<0.7,0.2>}	$\{\sim Salary_{20K}\}$

Let l be an element defined by a structured domain D_i. $U(e)$ is the set of higher level concepts, i.e. $U(e) = \{n | n \in D_i \wedge n$ is an ancestor of $l\}$, and $L(e)$ is the set of lower concepts $L(e) = \{n | n \in D_i \wedge n$ is a descendent of $l\}$. If l is a base concept then $L(e) = \varnothing$ and if l is a top level concept, then $U(e)=\varnothing$. Considering the H-IFS F={Research Associate/<1.0,0.0>, Programmer/<0.7,0.2>, Teaching Associate/<0.4,0.1>} as part of the Concept Employee in Fig.3:

$$U(\text{Programmer}/<0.7,0.2>) = \{ \text{Technical Staff}/<0.7,0.1>\}$$

$$L(\text{Programmer}/<0.7,0.2>) = \varnothing$$

Rule-1: If $(|U(e)| > 1 \wedge L(e) = \varnothing)$, then it is simply declared that a child or base concept has many parents.

E.g. $|U(\text{Programmer}/<0.7,0.2>)|=3$, $L(\text{Programmer}/<0.7,0.2>) = \varnothing$, Therefore a child or base concept acting as a selection predicate can claim any tuple (parent) containing elements found in $U(e)$, as its ancestor.

Now let us consider the following case where l_1="Employee" and l_2="Programmer" then let B the function that defines the space between $l_1 \wedge l_2$. In this case $|B((l_1),(l_2))|=2$, and >1

$B((l_1),(l_2)) = U(L(l_1) \wedge (l_2))$ where l_1 is a high level concept, l_2 is a base concept are elements defined in a structured domain. If both arguments are high level concepts or low level concepts then $B((l_1),(l_2)) = \varnothing$.

Rule-2: If $B((l_1),(l_2)$ is defined and $|B((l_1),(l_2))|>1$, then it is simply declared that multiple parents, high level concepts, are receiving a base concept as their own child. Therefore a parent or high level concept acting as a selection predicate can claim any tuple (child) containing elements found in $(L(l_1) \wedge (l_2))$, as its descendant, but with variants level of certainty.

Background knowledge may be specified as arising from a hierarchical Intuitionistic Fuzzy hierarchy, as integrity constraints, from the integration of conflicting databases, or from knowledge selected by domain experts. Using such information we offer to re-engineer the database by replacing missing, conflicting or unacceptable data by sets of the attribute domain.

Concept hierarchies have previously been used for attribute-induced knowledge discovery [18]. However the proposed use of background knowledge in this context is unique.

We assume that original attribute values may be given either as singleton sets, or subsets of the domain, or as concepts, which correspond to subsets of an attribute domain. In the last case the values may be defined in terms of a concept hierarchy. In addition there are rules describing the domain, and these may be formed in a number of ways: they may take the form of integrity constraints, where we have certain restrictions on domain values; functional dependencies and also rules specified by a domain expert.

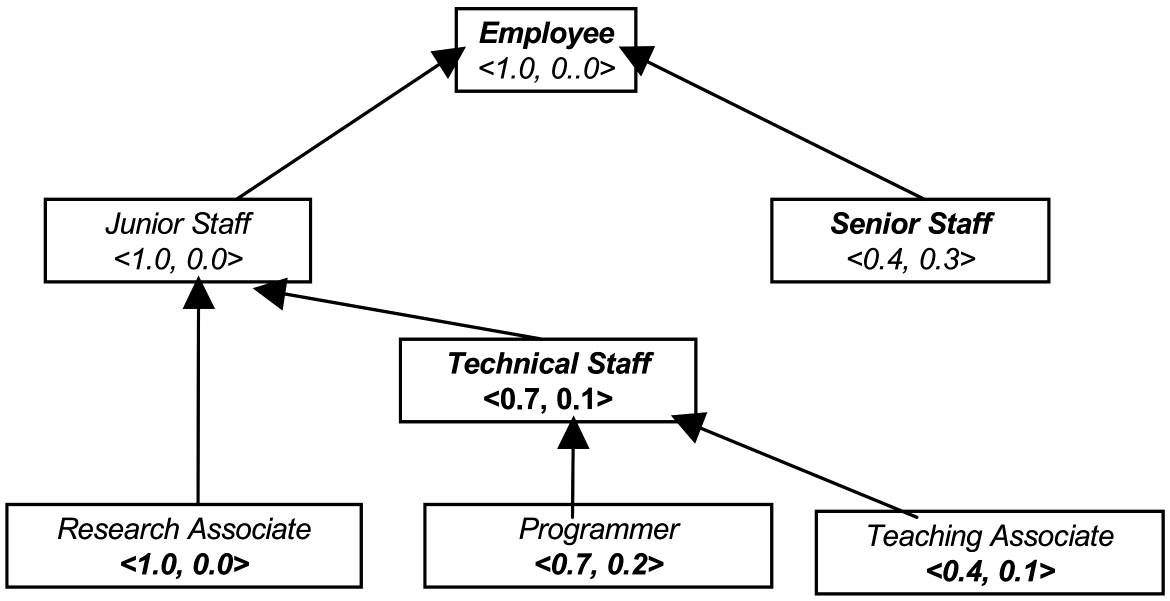


Fig. 3. Generalised H-IFS F –Concept Employee

An example of a concept hierarchy expressed with the aid of H-IFS F={Research Associate/<1.0,0.0>, Programmer/<0.7,0.2>, Teaching Associate/<0.4,0.1>} with values which are sets given in the generalised relation staff in Table.1. Here a value for the attribute JOB_DESCRIPTION may be a may be a concept from the concept hierarchy as defined in Fig.3 (e.g. {Technical Staff/ <0.7, 0.1>}). Then in terms of functional dependencies we may receive the following information Technical-Staff$\rightarrow$~Salary$_{25K}$. To this extent in terms of any declarative query language it can be concluded that the salary in for a reaching associate or a Programmer must be in the range of Salary$_{25K}$. We can also use this knowledge to approximate the salary for all instances of Junior Staff in case where no further background knowledge after estimating firstly the $<\mu,\nu>$ degrees for the hierarchical concept Employee. Such a hierarchical concept like employee in Fig.3 can be defined with the aid of Intuitionistic fuzzy set over a universe [19, 20] that has a hierarchical structure, named as H-IFS.

4 Definition of IFS and H-IFS

The notion of H-IFS rose from our need to express concepts [19] [20], [21], [22] in the case where these values are part of taxonomies as for food products or microorganisms for example .

The definition domains of the H-IFS sets that we propose below are subsets of hierarchies composed of elements partially ordered by the "kind of" relation. An element l_i is more general than an element l_j (denoted $l_i \sim l_j$), if l_i is a predecessor of l_j in the partial order induced by the "kind of" relation of the hierarchy. An example of such a hierarchy is given in Fig. 1. A hierarchical fuzzy set is then defined as follows.

Definition 4.1 A H-IFS is an Intuitionistic fuzzy set whose definition domain is a subset of the elements of a finite hierarchy partially ordered by the "kind of" $\leq$ relation.

For example, the fuzzy set M defined as: {Milk<0.8,0.1>, Whole-Milk<0.7,0.1>, Condensed-Milk<0.4,0.3>} conforms to Definition-3. Their definition domains are subsets of the hierarchy given in Fig.4.

We can note that no restriction has been imposed concerning the elements that compose the definition domain of a H-IFS. In particular, the user may associate a given $<\mu, \nu>$ with an element l_i and another degree $<\mu_1, \nu_1>$ with an element l_j more specific than l_i . $<\mu, \nu> \sim <\mu_1, \nu_1>$ represents a semantic of restriction for l_j compared to l_i, whereas $<\mu_1, \nu_1> \sim <\mu, \nu>$ represents a semantic of reinforcement for l_j compared to l_i. For example, if there is particular interest in condensed milk because the user studies the properties of low fat products, but also wants to retrieve complementary information about other kinds of milk, these preferences can be expressed using, for instance, the following Intuitionistic fuzzy set: <1, 0>/ condensed milk + <0.5, 0.1>/Milk. In this example, the element condensed milk has a greater degree than the more general element Milk, which corresponds to a semantic of reinforcement for condensed milk compared to Milk. We can make two observations concerning the use of H-IFSs:

- Let <1, 0>/ condensed milk + <0.5, 0.1>/Milk be an expression of liking in a query. We can note that this H-IFS implicitly gives information about elements of the hierarchy other than Condensed milk and Milk. One may also assume that any kind of condensed milk (i.e. whole condensed milk) interests the user with $<\mu, \nu> \rightarrow <1, 0>$.
- Two different H-IFSs on the same hierarchy do not necessarily have the same definition domain, which means they cannot be compared using the classic comparison operations of Intuitionistic fuzzy set theory For example, <1, 0>/ condensed milk + <0.5, 0.1>/Milk and 1/Milk + 0.2/ Pasteurised milk are defined on two different subsets of the hierarchy of "Fig. 1" and, thus, are not comparable.

These observations led to the introduction of the concept of closure of a Intuitionistic hierarchical fuzzy set, which is defined on the whole hierarchy. Intuitively, in the closure of a H-IFS, the "kind of, $\leq$" relation is taken into account by propagating the $<\mu, \nu>$ associated with an element to its sub-elements (more specific elements) in the hierarchy. For instance, in a query, if the user is interested in the element Milk, we consider that all kinds of Milk—Whole milk, Pasteurised milk, are also of interest. On the opposite, we consider that the super-elements (more general elements) of Milk in the hierarch are too broad to be relevant for the user's query.

Definition 4.2 Let F be a H-IFS defined on a subset D of the elements of a hierarchy L. It degree is denoted as $<\mu, \nu>$. The closure of F, denoted clos(F), is a H-IFS defined on the whole set of elements of L and its degree $<\mu, \nu>_{clos(F)}$ is defined as follows.

For each element **l** of L, let $S_L= \{l_1, \ldots, l_n\}$ be the set of the smallest super-elements of in D :

- **If S_L is not empty, $<\mu, \nu>_{clos(F)} (S_L) = <\max_{1\leq i\leq n}(\mu(L_i)), \min_{1\leq i\leq n}(\nu(L_i)>$**
  else, $<\mu, \nu>_{clos(F)} (S_L) = <0, 0>$

In other words, the closure of a H-IFS F is built according to the following rules. For each element l_I of L:

- If l_I belongs to F, then l_I keeps the same degree in the closure of F (case where $S_L= \{ l_I \}$).
- If l_I has a unique smallest super-element l_I in F, then the degree associated with l_I is propagated to L in the closure of F, $S_L= \{ l_I \}$ with $l_I > l_I$)

- if L has several smallest super-elements *{l_1, ...,l_n}* in F, with different degrees, a choice has to be made concerning the degree that will be associated with l_I in the closure. The proposition put forward in **Definition 4.2** consists of choosing the maximum degree of validity μ and minimum degree of non validity v associated with *{l_1, ...,l_n}*.
- All the other elements of L, i.e., those that are more general than, or not comparable with the elements of F, are considered as non-relevant. The degree <0,0> is associated with them.

If the H-IFS expresses preferences in a query, the choice of the maximum allows us not to exclude any possible answers. In real cases, the lack of answers to a query generally makes this choice preferable, because it consists of enlarging the query rather than restricting it.

If the H-IFS represents an ill formulated concept, the choice of the maximum allows us to preserve all the possible values of the datum, but it also makes the datum less specific. This solution is chosen in order to homogenize the treatment of queries and data. In a way, it enlarges the query, answer.

5 Replacing & Constraining Unknown Attribute Values

All descendents of an instance of a high-level concept are replaced with a **minimal H-IFS** has these descendents as members.

Defining the Minimal H-IFS

Step 1: Assign Min-H-IFS ← ∅. Establish an order so that the sub-elements {l_1,…,l_n} of the hierarchy L, are examined after its super-elements

Step 2: Let l_1 be the first element and $(l_1)/<\mu, \nu> \neq (l_1)/<0, 0>$ then add l_1 to Min-H-IFS and $<\mu, \nu>_{clos(Min\text{-}HIFS)}(l_1) = (l_1)/<\mu, \nu>$

Step 3: Let us assume that K elements of the hierarchy L satisfy the condition $<\mu, \nu>_{clos(Min\text{-}HIFS)}(l_i)=(l_i)/<\mu, \nu>$. In this case the Min-H-IFS do not change. Go to next element l_{k+1} and execute Step 4

Step 4: The $l_{k+1}/<\mu_{k+1}, \nu_{k+1}>$ associated with l_{k+1}. In this case l_{k+1} is added to Min-H-IFS with the corresponding $<\mu_{k+1}, \nu_{k+1}>$.

Step 5: Repeat steps three and four until $clos_{(Min\text{-}HIFS)}$=C

For instance the H-IFS's S_1 and S_2 are minimal (none of their elements is derivable). They cannot be reduced further.

S_1= Milk/<1,0>

S_2= {Milk/<1,0>, Whole-Milk/<0.7,0.1>, Whole-Pasteurised-milk/<1,0>, Condensed-Milk/<0.4, 0.3>}

In the next section, a complementary solution is proposed when it comes to lack of answers to a query, i.e. when the user wants to retrieve complementary answers close to his initial query. The H-IFS set that represents the user's preferences is replaced by a more general one

A **null** value is regarded as a partial value with all base domain values as members. We refer to the resultant partial value, obtained as a result of this process, as a **primal** partial value. The replacement process is thus performed by the following procedure:

Procedure: Replacement
Input: A concept table R consisting of partial values, or nulls.
Output: A re-engineered partial value table U.
Method: For each attribute value of R recursively replace the cell value by a **primal** partial value. For each cell of R replace, the primal partial value by a pruned prime-partial –value, until a minimal partial value is reached.

If a particular member of a partial value violates the domain constraint (rule) then it is **pruned** from the **minimal H-IFS primal partial value**. This process is continued until all partial values have been pruned by the constraints as much as possible. We refer to the resultant partial value, obtained as a result of this process, as a **minimal** partial value.

In addition in an integrated DBMS environment it will be also useful not to query all sources, but only those that contain information relevant to our request. This is quite critical for achieving better query performance. For this reason we equip our Integrated architecture with a repository that contains various constraints (i.e. Intuitionistic Fuzzy Range Constraints, Intuitionistic Fuzzy Functional Dependencies, etc) that are related to the information sources that participate in the Integrated Architecture.

Range constraints: such as "The average income per person is estimated to be in the range of €50K". Considering a finite universe of discourse, say X whose cardinality is N. Let us suppose that $X=\{X_1, X_2, \ldots, X_n\}$ and the Intuitionistic fuzzy number ~a given by ~a $=\{(x_i, \mu_i, \nu_i): x_i \in X, I = 1,2\ldots N\}$ We can express the above constraint as follows **~Income50K {(49, .8, .1), (50, .9, .02) (51, .7, .15)}**

Classical data integrity constraints such as "All persons stored at a source have a unique identifier".

Functional Dependencies: for instance, a source relation S1(Name, lives, income, Occupation) has a functional dependency Name$\rightarrow$(Lives, ~Income).These constraints are very useful to compute answers to queries.

There are several reasons we want to consider constraints separately from the query language. Describing constraints separately from the query language can allow us to do reasoning about the usefulness of a data source with respect to a valid user request.

Some of source constraints can be naturally represented as local constraints. Each **local constraint** is defined on one data source only. These constraints carry a rich set of semantics, which can be utilized in query processing. Any projected database instance of source, these conditions must be satisfied by the tuples in the database.

Definition 5.1 Let si,...,sl be l sources in a data-integrated system. Let P = {pi,..., pn } be a set of global predicates, on which the contents of each source s are defined. A general global constraint is a condition that should be satisfied by any database instance of the global predicates P.

General global constraints can be introduced during the design phase of such a data-integration system. That is, even if new sources join or existing ones leave the system, it is assumed that these constraints should be satisfied by any database instance of the global predicates. Given the global predicate **Income**, if a query asks for citizens with an average income above **~Income60K**, without checking the source contents and constraints, the integrated system can immediately know that the answer is empty.

To this extent we can interrogate the constraints repository to find out if a particular source contains relevant information with respect to particular request. We now consider the problem of aggregation for the partial value data model. In what follows we are concerned with symbolic attributes, which are typically described by counts and summarised by aggregated tables. The objective is to provide an aggregation operator which allows us to aggregate individual tuples to form summary tables.

6 Summary Tables and Aggregation

A summary table R, is represented in the form of an Intuitionistic fuzzy relation (IFR).

Aggregation (A): An aggregation operator A is a function $A(G)$ where $G = \{<x, \mu_F(x), \nu_F(x)> | x \in X\}$ where $x=<att_1, ..., att_n>$ is an ordered tuple belonging to a given universe X, $\{att_1, ..., att_n\}$ is the set of attributes of the elements of X, $\mu_F(x)$ *and* $\nu_F(x)$ are the degree of membership and non-membership of x. The result is a bag of the type $\{<x', \mu_F(x'), \nu_F(x')> | x' \in X\}$. To this extent, the bag is a group of elements that can be duplicated and each one has a degree of μ and ν.

Input: $R_i = (l, F, H)$ and the function $A(G)$

Output: $R_o = (l_o, F_o, H_o)$ *where*

- l is a set of levels $l_1, ..., l_n$, that belong to a partial order $\leq O$
  To identify the level l as part of a hierarchy we use dl.
  $l_\perp$: base level $l_\top$: top level
  for each pair of levels l_i and l_j we have the relation
  $\mu_{ij} : l_i \times l_j \rightarrow [0,1] \quad \nu_{ij} : l_i \times l_j \rightarrow [0,1] \quad 0 < \mu_{ij} + \nu_{ij} < 1$
- F is a set of fact instances with schema $F = \{<x, \mu_F(x), \nu_F(x)> | x \in X\}$, where $x=<att_1, ..., att_n>$ is an ordered tuple belonging to a given universe X, $\mu_F(x)$ *and* $\nu_F(x)$ are the degree of membership and non-membership of x in the fact table F respectively.
- H is an object type history that corresponds to a structure(l, F, H') which allows us to trace back the evolution of a structure after performing a set of operators i.e. aggregation

The definition of the extended group operators allows us to define the extended group operators ***Roll up (Δ), and Roll Down (Ω).***

Roll up (Δ): The result of applying Roll up over dimension d_i at level dl_r using the aggregation operator A over a relation $R_i=(l_i, F_i, H_i)$ is another relation $R_o=(l_o, F_o, H_o)$

Input: $R_i = (l_i, F_i, H_i)$

Output: $R_o = (l_o, F_o, H_o)$

An object of type history is a recursive structure $H = \begin{cases} \omega \text{ is the initial state of the relation.} \\ (l, A, H') \text{ is the state of the relation after performing an operation on it.} \end{cases}$

The structured history of the relation allows us to keep all the information when applying *Roll up* and get it all back when *Roll Down* is performed. To be able to apply the operation of *Roll Up* we need to make use of the IF_{SUM} aggregation operator.

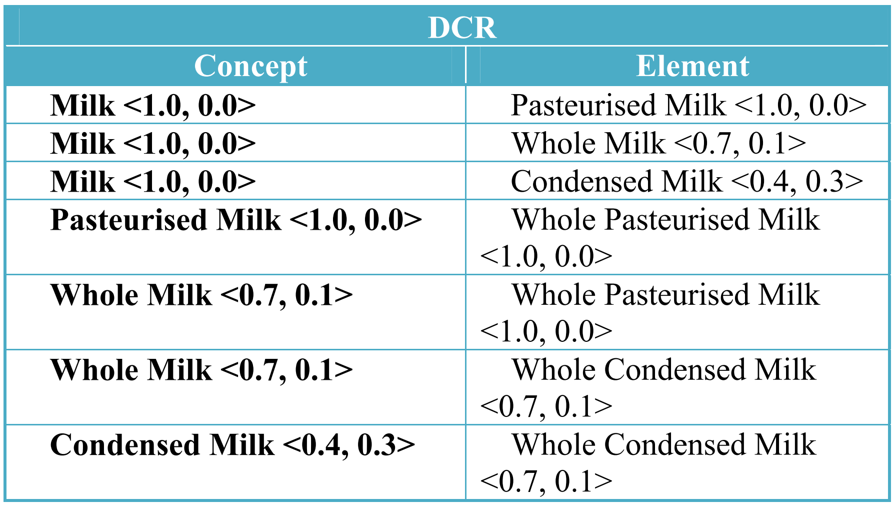
Roll Down (Ω): This operator performs the opposite function of the *Roll Up* operator. It is used to roll down from the higher levels of the hierarchy with a greater degree of generalization, to the leaves with the greater degree of precision. The result of applying *Roll Down* over a relation $R_i = (l, F, H)$ having $H=(l', A', H')$ is another relation $R_o = (l', F', H')$.
Input: $R_i=(l, F, H)$
Output: $R_o=(l', F', H')$ where F' → *set of fact instances defined by operator A.*
To this extent, the *Roll Down* operative makes use of the recursive history structure previously created after performing the *Roll Up* operator.

6.1 Summarisation Paths

The structure of any H-IFS can be described by a domain concept relation DCR = (Concept, Element), where each tuple describes a relation between elements of the domain on different levels.
The DCR can be used in calculating recursively the different summarisation or selection paths as follows:

$$\text{PATH} \leftarrow \text{DCR} \underset{\{x=1\ldots(n-2)\,|\,n>2\}}{\bowtie} \text{DCR}_x$$

If n≤2, then DCR becomes the Path table as it describes all summarisation and selection paths. These are entries to a knowledge table that holds the metadata on parent-child relationships. An example is presented below:



DCR	
Concept	**Element**
Milk <1.0, 0.0>	Pasteurised Milk <1.0, 0.0>
Milk <1.0, 0.0>	Whole Milk <0.7, 0.1>
Milk <1.0, 0.0>	Condensed Milk <0.4, 0.3>
Pasteurised Milk <1.0, 0.0>	Whole Pasteurised Milk <1.0, 0.0>
Whole Milk <0.7, 0.1>	Whole Pasteurised Milk <1.0, 0.0>
Whole Milk <0.7, 0.1>	Whole Condensed Milk <0.7, 0.1>
Condensed Milk <0.4, 0.3>	Whole Condensed Milk <0.7, 0.1>

Fig. 4. Domain Concept Relation

Fig. 5 shows how our Milk hierarchy knowledge table is kept. Paths are created by running a recursive query that reflects the 'PATH' algebraic statement. The hierarchical IFS used as example throughout this paper comprises of 3 levels, thus calling for the SQL-like query as below:

SELECT A.Concept as Grand-concept, b.concept, b.element
FROM DCR as A, DCR as B
WHERE A.child=B.parent;

This query will produce the following paths:

Path			
Grand-concept	**Concept**	**Element**	**Path Colour**
Milk <1.0, 0.0>	Pasteurised Milk <1.0, 0.0>	Whole Pasteurised Milk <1.0, 0.0>	Red
Milk <1.0, 0.0>	Whole Milk <0.7, 0.1>	Whole Pasteurised Milk <1.0, 0.0>	Blue
Milk <1.0, 0.0>	Whole Milk <0.7, 0.1>	Whole Condensed Milk <0.7, 0.1>	Green
Milk <1.0, 0.0>	Condensed Milk <1.0, 0.0>	Whole Condensed Milk <0.7, 0.1>	Brown

Fig. 5. Path Table

Fig.7 presents a pictorial view of the four distinct summarisation and selection paths.

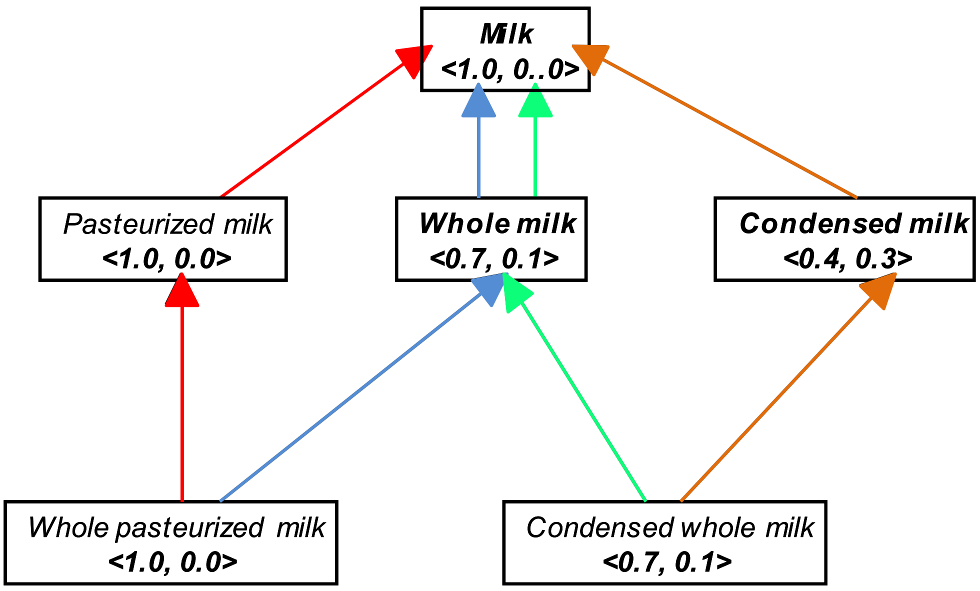


Fig. 6. Pictorial representation of paths

These paths will be used in fuzzy queries to extract answers that could be either definite or possible. This will be realised with the aid of the predicate (θ).

A predicate (θ) involves a set of atomic predicates ($\theta_1, \ldots, \theta_n$) associated with the aid of logical operators *p (* i.e. $\wedge$, $\vee$, etc.). Consider a predicate θ that takes the value "Whole Milk", θ = "Whole Milk".

After utilizing the IFS hierarchy presented in Fig.7, this predicate can be reconstructed as follows:

$$\theta = \theta_1 \vee \theta_2 \vee \ldots \vee \theta_n$$

In our example, θ_1="Whole Milk", θ_2="Whole Pasteurised Milk" and θ_n="Condensed Whole Milk".

The reconstructed predicate θ = (Whole Milk $\vee$ Whole Pasteurised Milk $\vee$ Condensed Whole Milk) allows the query mechanism to not only definite answers, but also possible answers.

In terms a query retrieving data from a summary table, the output contains not only records that match the initial condition, but also those that satisfy the reconstructed predicate. Consider the case where no records satisfy the initial condition (Whole Milk). Traditional aggregation query would have returned no answer, however, based on our approach, the extended query would even in this case, return an answer, though only a possible one, with a specific belief and disbelief $<\mu, \nu>$. It will point to those records that satisfy the reconstructed predicateθ, more specifically, "Whole Pasteurised Milk and Condensed Whole Milk".

7 Conclusions

We provide a means of using background knowledge to re-engineer the data representation into a partial value representation with the aid of H-IFS and Intuitionistic Fuzzy relational representation.

The hierarchical links are defined by the "kind of, $\leq$" relation. The membership of an element in a H-IFS has consequences on the membership and non-membership of its sub elements in this set. The notion of H-IFS, that may be defined on a part of a hierarchy and the notion of closure of a H-IFS, that is explicitly defined on the whole hierarchy, using the links between the elements that compose the hierarchy.

H-IFSs that have the same closure define equivalence classes, called minimal H-IFS. Minimal fuzzy sets are used as a basis to define the generalization of a H-IFS fuzzy set. The proposed methodology aims at enlarging the user preferences expressed when defining a query, in order to obtain related and complementary answers.

We have discussed how domain knowledge presented in the form of background knowledge, such as integrity constraints, functional dependencies or details of the concept hierarchy, may be used to reduce the amount of missing data in the database..

We have presented a new multidimensional model that is able to operate over data with imprecision in the facts and the summarisation hierarchies. Classical models imposed a rigid structure that made the models present difficulties when merging information from different but still reconcilable sources.

This is likely to be a useful tool for decision support and knowledge discovery in, for example, data mediators, data warehouses, where the data are often subject to such imperfections. Furthermore we notice that our approach can be used for the representation of Intuitionistic fuzzy linguistic terms

References

[1] Bell, D., Guan, J., Lee, S.: Generalized union and project operations for pooling uncertain and imprecise information. DKE 18 (1996) 89-117

[2] Chen, A., Tseng, F.: Evaluating aggregate operations over imprecise data. IEEE Trans. on Knowledge and Data Engineering, 8 (1996) 273-284

[3] Zemankova, M., Kandel, A.: Implementing imprecision in Information systems. Inf. Sci. Vol. 37. (1985) 107–141

[4] Dubois, D., Prade, H., Testamale, C.: Handling Incomplete or Uncertain Data and Vague Queries in Database Applications. Plenum Press (1988)

[5] Prade, H.: Annotated bibliography on fuzzy information processing. Readings on Fuzzy Sets in Intelligent Systems. Morgan Kaufmann Publishers Inc. (1993)

[6] Codd, E.: Extending the Data Base Relational Model to Capture More Meaning. ACM Trans. Database Systems, Vol.4, No.4, 397-434, 1979

[7] Goldstein, B.: Constraints on Null Values in Relational Databases. Proc. 7th Int. Conf. on VLDB, IEEE Press (1981) 101-110

[8] Biskup, J.: A Foundation of Codd's Relational Maybe-Operations. XP2 Workshop on Relational Database Theory. (1981)

[9] Liu, C., Sunderraman, R.: Indefinite and maybe information in relational databases. ACM Trans. Database Syst, Vol.15, No.1 (1990) 1–39.

[10] Liu, K., Sunderraman, R.: On Representing Indefinite and Maybe Information in Relational Databases: A Generalization. ICDE. IEEE Computer Society (1990) 495-502

[11] Ola, A.: Relational databases with exclusive disjunctions. Data Engineering (1992) 328–336

[12] Homenda, W.: Databases with Alternative Information. IEEE Trans. on Knowledge and Data Engineering, Vol. 3, No. 3. (1991) 384-386.

[13] Gessert, G.: Handling Missing Data by Using Stored Truth Values. SIGMOD Record, Vol. 20, No. 1 (1991) 30-42

[14] Zicari, R.: Closed World Databases Opened Through Null Values. Proc. 14th Int. Conf. on VLDB. (1988) 50-61

[15] Zaniolo, C.: Database relations with null values. J. Comput. Syst. Sci. Vol.28 (1984) 142–166.

[16] Lipski, J.: On semantic issues connected with incomplete information databases. ACM Trans. Database Syst, Vol.4, No.3 (1979) 262–296

[17] Dhar, V., Tuzhilin, A.: Abstract-driven pattern discovery in databases. IEEE Trans. on Knowledge and Data Engineering 6 (1993) 926-938

[18] Han, J., Fu, Y.: Attribute-oriented induction in data mining. Advances in Knowledge Discovery. AAAI Press/MIT Press, Cambridge, MA (1996) 399-421

[19] Rogova E., Chountas P., Atanassov, K.: Flexible Hierarchies and Fuzzy Knowledge-based OLAP. FSKD 2007,

[20] Rogova E., Chountas P.: On imprecision intuitionistic fuzzy sets & OPLAP – The case for KNOLAP. IFSA 2007

[21] Atanassov, K.: Intuitionistic Fuzzy Sets. Springer-Verlag. Heidelberg. (1999)

[22] Atanassov, K.: Intuitionistic Fuzzy Sets, Fuzzy Sets and Systems. (1986) 20, 87–96