A Certainty-Based Approach to the Cautious Handling of Suspect Values

Olivier Pivert and Henri Prade

Abstract In this paper, we consider the situation where a database may contain suspect values, i.e. precise values whose validity is not certain but whose attached uncertainty level is unknown. We propose a database model based on the notion of possibilistic certainty to deal with such values. The operators of relational algebra are extended in this framework. A crucial aspect is that queries have the same data complexity as in a classical database context.

1 Introduction

The need to handle uncertain values in a database context has been recognized a long time ago, and a variety of uncertain database models have been proposed to this aim. In these models, an ill-known attribute value is generally represented by a probability distribution (see, e.g. [4, 10]) or a possibility distribution [1], i.e. a set of weighted candidate values. However, in many situations, it may be very problematic to quantify the level of uncertainty attached to the different candidate values. One may not even know the set of (probable/possible) alternative candidates. Then, using a probabilistic model in a rigorous manner appears quite difficult, not to say impossible. In this work, we assume that all one knows is that a given precise value is suspect, i.e. not totally certain, and we show that a database model based on the notion of possibilistic certainty is a suitable tool for representing and handling suspect data. An introductory and abridged version can be found in [8].

O. Pivert(✉)
Irisa, University of Rennes 1, Lannion, France
e-mail: pivert@enssat.fr

H. Prade
IRIT–CNRS, University of Toulouse 3, Toulouse, France
e-mail: prade@irit.fr

© Springer International Publishing Switzerland 2016

T. Andreasen et al. (eds.), *Flexible Query Answering Systems 2015*,
Advances in Intelligent Systems and Computing 400,
DOI: 10.1007/978-3-319-26154-6_6

The remainder of the paper is structured as follows. Section 2 presents the uncertain database model that we advocate for representing tuples that may i) involve suspect attribute values, ii) be themselves uncertain (due to the fact that some operations generate "maybe tuples"). Section 3 gives the definitions of the algebraic operators in this framework. Query equivalences are studied in Section 4. In Section 5, we discuss a way to make selection queries more flexible, which makes it possible to discriminate the uncertain answers to a query. Finally, Section 6 recalls the main contributions and outlines perspectives for future work.

2 The Model

In possibility theory [3, 11], each event E — defined as a subset of a universe Ω — is associated with two measures, its possibility $\Pi(E)$ and its necessity $N(E)$. Π and N are two dual measures, in the sense that $N(E) = 1 - \Pi(\overline{E})$ (where the overbar denotes complementation). This clearly departs from the probabilistic situation where $Prob(E) = 1 - Prob(\overline{E})$. In possibility theory, being somewhat certain about E ($N(E)$ has a high value) forces you to have $\overline{E}$ rather impossible ($1 - \Pi$ is impossibility), but it is allowed to have no certainty neither about E nor about $\overline{E}$. Generally speaking, possibility theory is oriented towards the representation of epistemic states of information, while probabilities are deeply linked to the ideas of randomness, and of betting in case of subjective probability.

In the following, we assume that the certainty degree associated with the uncertain events considered (that concern the actual value of an attribute in a tuple, for instance) is unknown. Thus, we use a fragment of possibility theory where three values only are used to represent certainty : 1 (completely certain), α (somewhat certain but not totally), 0 (not at all certain). The fact that one uses α for every somewhat certain event does not imply that the certainty degree associated with these events is the same; α is just a conventional symbol that means "a certainty degree in the open interval $(0, 1)$". Notice that this corresponds to using three symbols for representing possibility degrees as well: 0, β $(= 1 - \alpha)$, and 1 (but we are not interested in qualifying possibility here). In other words, we are representing pieces of information of the form $N(E) \geq \alpha$, which might be regarded as "known unknowns" [5] in the sense that their certainty level α cannot be precisely assessed, while "unknown unknowns" remain out of reach (states of acknowledged ignorance corresponding to $\Pi(E) = \Pi(\overline{E}) = 1$, or equivalently $N(E) = N(\overline{E}) = 0$, cannot be captured either).

The model that we introduce hereafter is a simplified — thus "lighter" — version of that introduced in [2] and detailed in [9], where a certainty level is attached to each ill-known attribute value (by default, an attribute value has certainty 1). As we will see, representing suspect attribute values also leads us to representing the fact that the existence of some tuples (in the result of some queries) may not be totally certain either. Let us first discuss the philosophy of the approach, before describing the model and the operations more precisely.

Let us consider a database containing suspect values. In the following, a suspect value will be denoted using a star, as in 17^*. A value a^* means that it is somewhat

certain (thus completely possible) that a is the actual value of the considered attribute for the considered tuple, but not totally certain (otherwise we would use the notation a instead of a^*). When evaluating a selection query Q based on a condition ψ (made of atomic conjuncts ψ_i), one may distinguish between three groups of answers:

- the *completely certain* answers. A tuple t is a completely certain answer to Q iff t does not contain any suspect value concerned by ψ, and every attribute value of t concerned by a conjunct ψ_i satisfies ψ_i. For instance, $t = \langle$John, 35*, Paris$\rangle$ is a completely certain answer to $Q = \sigma_{city='Paris'}(r)$.
- the *somewhat certain* (thus completely possible) answers. A tuple t is a somewhat certain answer to Q iff i) t contains at least one suspect value concerned by a conjunct ψ_i, and every suspect value concerned by a conjunct ψ_i satisfies this conjunct, ii) every nonsuspect attribute value of t concerned by a conjunct ψ_i satisfies ψ_i. For instance, $t = \langle$John, 35*, Paris$\rangle$ is a somewhat certain answer to $Q = \sigma_{city='Paris' \text{ and } age=35}(r)$.
- the *somewhat possible* (but not at all certain) answers. A tuple t is a somewhat possible answer iff i) t contains at least one suspect value that does not satisfy the corresponding conjunct ψ_i from ψ, ii) every nonsuspect attribute value of t concerned by a conjunct ψ_i satisfies ψ_i. For instance, $t = \langle$John, 35*, Paris$\rangle$ is a somewhat possible answer to $Q = \sigma_{city='Paris' \text{ and } age=40}(r)$.

As to the other tuples (those that contain at least one nonsuspect value that does not satisfy the associated conjunct ψ_i from ψ), they are of course discarded.

In fact, in the model we propose, we restrict ourselves to the computation of the two first groups (i.e., the completely or somewhat certain answers), since dealing with the answers that are only somewhat possible raises important difficulties. Namely, in order to have a sound compositional framework (and to preserve the possible worlds semantics), one would have to be able to represent not only values of the type a^* but one would need to maintain a complete representation of attribute values in terms of possibility distributions. Moreover, we are then faced with the problem of handling intertuple dependencies generated by the join operation in particular [9].

Notice that the framework we propose is compatible with the use of NULL values for representing attribute values that exist but are currently (totally) unknown. If a tuple includes a NULL for an attribute concerned by a selection condition, it will simply be discarded since we are only interested in answers that are somewhat certain.

Uncertain tuples are denoted by α/t where α has the same meaning as above. α/t means that the existence of the tuple in the considered relation is only somewhat certain (thus, it is also possible to some extent that it does not exist). It is mandatory to have a way to represent such uncertain tuples since some operations of relational algebra (selection, in particular) may generate them. The tuples whose existence is completely certain are denoted by $1/t$. A relation of the model will thus involve an extra column denoted by N, representing the certainty attached to the tuples.

3 Algebraic Operators

In this section, we give the definition of the different operators of relational algebra in the certainty-based model defined above.

3.1 Selection

Let us denote by $c(t.A)$ the certainty degree associated with the value of attribute A in tuple t: $c(t.A)$ equals 1 if $t.A$ is a nonsuspect value, and it takes the (conventional) value α otherwise (with $\alpha < 1$). It is the same thing for the certainty degree N associated with a tuple (the notation is then N/t).

Case of a condition of the form $A\ \theta\ q$ where A is an attribute, θ is a comparison operator, and q is a constant:

$$\sigma_{A\,\theta\,q}(r) = \{N'/t \mid N/t \in r \text{ and } t.A\ \theta\ q \text{ and } N' = \min(N,\ c(t.A))\} \tag{1}$$

Example 1. Let us consider the relation *Emp* represented in Table 1 (left) and the selection query $\sigma_{job='Engineer'}(Emp)$. Its result appears in Table 1 (right). ⋄

Table 1 Relation *Emp* (left), result of the selection query (right)

#id	*name*	*city*	*job*	*N*
37	John	Newton*	Engineer*	1
53	Mary	Quincy*	Clerk*	1
71	Bill	Boston	Engineer	1

#id	*name*	*city*	*job*	*N*
37	John	Newton*	Engineer*	α
71	Bill	Boston	Engineer	1

Case of a condition of the form $A_1\ \theta\ A_2$ where A_1 and A_2 are two attributes and θ is a comparison operator:

$$\begin{aligned}\sigma_{A_1\,\theta\,A_2}(r) = \{N'/t \mid N/t \in r \text{ and } t.A_1\ \theta\ t.A_2 \text{ and} \\ N' = \min(N,\ c(t.A_1),\ c(t.A_2))\}.\end{aligned} \tag{2}$$

Case of a conjunctive condition $\psi = \psi_1 \wedge \ldots \wedge \psi_m$:

$$\begin{aligned}\sigma_{\psi_1 \wedge \ldots \wedge \psi_m}(r) = \{N'/t \mid N/t \in r \text{ and } \psi_1(t.A_1) \text{ and } \ldots \text{ and } \psi_m(t.A_m) \\ \text{and } N' = \min(N,\ c(t.A_i),\ \ldots,\ c(t.A_m))\}.\end{aligned} \tag{3}$$

Case of a disjunctive condition $\psi = \psi_1 \vee \ldots \vee \psi_m$:

$$\begin{aligned}\sigma_{\psi_1 \vee \ldots \vee \psi_m}(r) = \{N'/t \mid N/t \in r \text{ and } (\psi_1(t.A_1) \text{ or } \ldots \text{ or } \psi_m(t.A_m)) \\ \text{and } N' = \min(N, \max_{i \text{ such that } \psi_i(t.A_i)} (c(t.A_i)))\}.\end{aligned} \tag{4}$$

3.2 Projection

Let r be a relation of schema (X, Y). The projection operation is straightforwardly defined as follows:

$$\pi_X(r) = \{N/t.X \mid N/t \in r \text{ and } \nexists\, N'/t' \in r \text{ such that } sbs(N'/t'.X,\ N/t.X)\}.$$

The only difference w.r.t. the definition of the projection in a classical database context concerns duplicate elimination, which is here based on the concept of "possibilistic subsumption". Let $X = \{A_1, \ldots, A_n\}$. The predicate sbs, which expresses subsumption, is defined as follows:

$$\begin{aligned} sbs((N'/t'.X, N/t.X) \equiv \\ &\forall i \in \{1, \ldots, n\},\ t.A_i = t'.A_i \text{ and} \\ &c(t.A_i) \leq c(t'.A_i) \text{ and } N \leq N' \text{and} \\ &((\exists i \in \{1, \ldots, n\},\ c(t.A_i) < c(t'.A_i)) \text{ or } N < N'). \end{aligned} \tag{5}$$

Example 2. Let us consider relation *Emp* represented in Table 2 (left) and the projection query $\pi_{\{city,\ job\}}$ (*Emp*). Its result is represented in Table 2 (right). ⋄

Table 2 Relation *Emp* (left), result of the projection query (right)

#id	*name*	*city*	*job*	*N*
35	Phil	Newton	Engineer*	1
52	Lisa	Quincy*	Clerk*	α
71	Bill	Newton	Engineer	α
73	Bob	Newton*	Engineer*	α
84	Jack	Quincy*	Clerk	α

city	*job*	*N*
Newton	Engineer*	1
Newton	Engineer	α
Quincy*	Clerk	α

3.3 Join

The definition of the join in the context of the model considered is:

$$\begin{aligned} r_1 \bowtie_{A=B} r_2 = \{ &\min(N_1,\ N_2,\ c(t_1.A),\ c(t_2.B))/t_1 \oplus t_2 \mid \\ &\exists\, N_1/t_1 \in r_1,\ \exists\, N_2/t_2 \in r_2 \text{ such that } t_1.A = t_2.B \end{aligned} \tag{6}$$

where $\oplus$ denotes concatenation.

Example 3. Consider the relations *Person* and *Lab* from Table 3 and the query:

$$PersLab = Person \bowtie_{Pcity=Lcity} Lab$$

Table 3 Relations *Person* (left), *Lab* (right), result of the join query (bottom)

#Pid	*Pname*	*Pcity*	*N*
11	John	Boston*	1
12	Mary	Boston	α
17	Phil	Weston*	α
19	Jane	Weston	1

#Lid	*Lname*	*Lcity*	*N*
21	BERC	Boston*	α
22	IFR	Weston	1
23	AZ	Boston	1

#Pid	*Pname*	*Pcity*	*#Lid*	*Lname*	*Lcity*	*N*
11	John	Boston*	21	BERC	Boston*	α
11	John	Boston*	23	AZ	Boston	α
12	Mary	Boston	21	BERC	Boston*	α
12	Mary	Boston	23	AZ	Boston	α
17	Phil	Weston*	22	IFR	Weston	α
19	Jane	Weston	22	IFR	Weston	1

which looks for the pairs (p, l) such that p (somewhat certainly) lives in a city where (somewhat certainly) a research center l is located. Its result appears in Table 3 (bottom). $\diamond$

In the case of a natural join (i.e., an equijoin on all of the attributes common to the two relations), one keeps only one copy of each join attribute in the resulting table. Here, this "merging" keeps the more uncertain value for each join attribute. This behavior is illustrated in Table 4.

Table 4 Result of the natural join query (assuming a common attribute *City*)

#Pid	*Pname*	*City*	*#Lid*	*Lname*	*N*
11	John	Boston*	21	BERC	α
11	John	Boston*	23	AZ	α
12	Mary	Boston*	21	BERC	α
12	Mary	Boston	23	AZ	α
17	Phil	Weston*	22	IFR	α
19	Jane	Weston	22	IFR	1

3.4 Intersection

For the sake of readability of the following definition, we denote a suspect value v^* by $(v,\ \alpha)$ and a totally certain value v by $(v,\ 1)$. The intersection $r_1 \cap r_2$ is defined as follows:

$$
\begin{aligned}
&r_1 \cap r_2 = \\
&\quad \{\min(N_1,\ N_2)/\langle(t.A_1,\ \min(\rho_{1,1},\ \rho_{2,1})),\ \ldots,\ (t.A_n,\ \min(\rho_{1,n},\ \rho_{2,n}))\rangle \\
&\qquad \text{such that } N_1/\langle(t.A_1,\ \rho_{1,1}),\ldots,\ (t.A_n,\ \rho_{1,n})\rangle \in r_1 \\
&\qquad \text{and } N_2/\langle(t.A_1,\ \rho_{2,1}),\ldots,\ (t.A_n,\ \rho_{2,n})\rangle \in r_2\}.
\end{aligned} \tag{7}
$$

Recall that $\rho_{i,j}$ equals either 1 (certain value) or α (suspect value).

Example 4. Consider the relations from Table 5 (left and middle). The result of the intersection query $r_1 \cap r_2$ is given in Table 5 (right). $\diamond$

Table 5 Relations r_1 (left), r_2 (middle), and $r_1 \cap r_2$ (right)

Job	*City*	*N*
Engineer*	Boston	1
Engineer	Newton*	α
Technician	Boston	1
Technician	Quincy*	1

Job	*City*	*N*
Engineer*	Boston*	1
Engineer	Newton	1
Clerk*	Boston	α
Technician	Boston	1

Job	*City*	*N*
Engineer*	Boston*	1
Engineer	Newton*	α
Technician	Boston	1

3.5 Union

Union is defined as usual (and has the same data complexity), except that duplicate elimination is based on the notion of "possibilistic subsumption" (see Subsection 3.2) :

$$\begin{aligned} & r_1 \cup r_2 = \\ & \{N/t \mid (N/t \in r_1 \text{ and } N'/t \notin r_2) \text{ or } (N/t \in r_2 \text{ and } N'/t \notin r_1) \text{ or} \\ & \quad (N/t \in r_1 \text{ and } N/t \in r_2 \text{ or} \\ & \quad (N/t \in r_1 \text{ and } N'/t' \in r_2 \text{ and } V(t) = V(t') \text{ and } sbs(N/t,\ N'/t')) \text{ or} \\ & \quad (N/t \in r_2 \text{ and } N'/t' \in r_1 \text{ and } V(t) = V(t') \text{ and } sbs(N/t,\ N'/t'))\} \end{aligned} \tag{8}$$

where $V(t)$ is the tuple formed with the attribute values of t made certain. For instance, if $t = \langle 17, \text{John}, \text{Engineer*}, \text{Boston}, 35* \rangle$, then $V(t) = \langle 17, \text{John}, \text{Engineer}, \text{Boston}, 35 \rangle$.

Example 5. Consider the relations from Table 6 (left and middle). The result of the union query $r_1 \cup r_2$ is given in Table 6 (right). $\diamond$

3.6 Difference

Let us now consider the difference $r_1 - r_2$. Its definition is as follows:

$$\begin{aligned} r_1 - r_2 = \{N'/t \mid N/t \in r_1 \text{ and } N' = \min(N,\ \delta) \text{ with} \\ \delta = \min_{t' \in r_2} \max_{i=1..n} N(t.A_i \neq t'.A_i)\} \end{aligned} \tag{9}$$

where

$$N(t.A_i \neq t'.A_i) = \begin{cases} 1 \text{ if } t.A_i \neq t'.A_i \text{ and } c(t.A_i) = c(t'.A_i) = 1, \\ 0 \text{ if } t.A_i = t'.A_i, \\ \alpha \text{ otherwise.} \end{cases}$$

Table 6 Relations r_1 (left), r_2 (middle), and $r_1 \cup r_2$ (right)

Job	*City*	*N*
Engineer*	Boston	1
Engineer*	Quincy*	α
Manager	Newton	1

Job	*City*	*N*
Manager*	Boston	α
Manager	Newton	1
Engineer*	Boston*	α

Job	*City*	*N*
Engineer*	Boston	1
Engineer*	Quincy*	α
Manager	Newton	1
Manager*	Boston	α

In Formula 9, δ corresponds to the certainty degree associated with the event "all the tuples t' from r_2 differ from t" or, in other words, "for every tuple t' that appears in r_2, one of the attribute values of t' differs from the corresponding attribute value in t". The universal quantifier ("for every tuple") is interpreted by a min whereas the existential one ("one of the attribute values") is interpreted by a max.

Example 6. Consider the relations from Table 5 (left and middle). The result of $r_1 - r_2$ is given in Table 7 (right). Note that the tuple ⟨Manager, Chicago⟩ is uncertain in the result because of the tuple ⟨Cashier*, Boston*⟩ from r_2 which has a $(1-\alpha)$-possible interpretation equal to ⟨Manager, Chicago⟩. ◇

Table 7 Relations r_1 (left), r_2 (middle), and $r_1 - r_2$ (right)

Job	*City*	*N*
Engineer*	Boston*	1
Engineer	Quincy*	1
Manager	Chicago	1
Clerk	Springfield	1

Job	*City*	*N*
Engineer	Newton*	α
Clerk	Quincy*	α
Clerk	Springfield	α
Cashier*	Boston*	α

Job	*City*	*N*
Engineer*	Boston*	α
Engineer	Quincy*	α
Manager	Chicago	α

A crucial point is that the join operation does not induce intertuple dependencies in the result, due to the semantics of certainty. This is not the case when a probabilistic or a full possibilistic [1] model is used, and one then has to use a variant of c-tables [6] to handle these dependencies, which implies a non-polynomial complexity. On the other hand, since none of the operators of relational algebra induces intertuple dependencies in our model, the queries have the same data complexity as in a classical database context; see [9] for a more complete discussion.

4 About Query Equivalences

Let us recall that relational algebraic queries can be represented as a tree where the internal nodes are operators, leaves are relations, and subtrees are subexpressions. The primary goal of query optimization is to transform expression trees into equivalent ones, where the average size of the relations yielded by subexpressions in the tree are smaller than they were before the optimization. This transformation process uses a set of properties (query equivalences), and the question arises whether these properties

remain valid in the certainty-based model. The most common query equivalences are:

1. $\pi_X(\pi_{XY}(r)) = \pi_X(r)$,
2. $\sigma_{\psi_2}(\sigma_{\psi_1}(r)) = \sigma_{\psi_1}(\sigma_{\psi_2}(r)) = \sigma_{\psi_1 \wedge \psi_2}(r)$,
3. $\sigma_{\psi_1 \vee \psi_2}(r) = \sigma_{\psi_1}(r) \cup \sigma_{\psi_2}(r)$,
4. $\sigma_{\psi}(\pi_X(r)) = \pi_X(\sigma_{\psi}(r))$ if ψ concerns X only,
5. $\sigma_{\psi}(r_1 \times r_2) = \sigma_{\psi_1 \wedge \psi_2 \wedge \psi_3}(r_1 \times r_2) = \sigma_{\psi_3}(\sigma_{\psi_1}(r_1) \times \sigma_{\psi_2}(r_2))$ where $\psi = \psi_1 \wedge \psi_2 \wedge \psi_3$ and ψ_1 concerns only attributes from r_1, ψ_2 concerns only attributes from r_2, and ψ_3 is the part of ψ that concerns attributes from both r_1 and r_2,
6. $\sigma_{\psi}(r_1 \cup r_2) = \sigma_{\psi}(r_1) \cup \sigma_{\psi}(r_2)$,
7. $\sigma_{\psi}(r_1 \cap r_2) = \sigma_{\psi}(r_1) \cap \sigma_{\psi}(r_2) = \sigma_{\psi}(r_1) \cap r_2 = r_1 \cap \sigma_{\psi}(r_2)$,
8. $\sigma_{\psi}(r_1 - r_2) = \sigma_{\psi}(r_1) - \sigma_{\psi}(r_2) = \sigma_{\psi}(r_1) - r_2$,
9. $\pi_X(r_1 \cup r_2) = \pi_X(r_1) \cup \pi_X(r_2)$,
10. $\pi_Z(r_1 \times r_2) = \pi_X(r_1) \times \pi_Y(r_2)$ if X (resp. Y) denotes the subset of attributes of Z present in r_1 (resp. r_2).

It is straightforward to prove that all of these equivalences remain valid in the model we propose (they are direct consequences of the definitions of the operators given above). As an illustration, let us demonstrate Property 3.

Proof. Let us assume that condition ψ_1 (resp. ψ_2) concerns attribute A_1 (resp. A_2). Let us consider a tuple N/t from r such that $\psi_1(t.A_1) \vee \psi_2(t.A_2)$ holds (which is a necessary condition for t to belong to $(\sigma_{\psi_1 \vee \psi_2}(r))$ on the one hand, and to $(\sigma_{\psi_1}(r) \cup \sigma_{\psi_2}(r)))$ on the other hand. Four cases have to be considered:

- $c(t.A_1) = 1$ and $c(t.A_2) = 1$: then, N/t generates a tuple N/t in $\psi_1(t.A_1) \vee \psi_2(t.A_2)$ (indeed $\min(N, \max(1, 1)) = \min(N, 1) = N$) and a tuple N/t both in σ_{ψ_1} and in σ_{ψ_2}, thus in $(\sigma_{\psi_1}(r) \cup \sigma_{\psi_2}(r))$.
- $c(t.A_1) = 1$ and $c(t.A_2) = \alpha$: then, N/t generates a tuple N/t in $\psi_1(t.A_1) \vee \psi_2(t.A_2)$ (indeed $\min(N, \max(1, \alpha)) = \min(N, 1) = N$). It also generates a tuple N/t in σ_{ψ_1} and a tuple α/t in σ_{ψ_2}. Thus, it generates a tuple N/t in $(\sigma_{\psi_1}(r) \cup \sigma_{\psi_2}(r))$ since N/t subsumes (or is equal to) α/t.
- $c(t.A_1) = \alpha$ and $c(t.A_2) = 1$: this case is similar to the previous one.
- $c(t.A_1) = \alpha$ and $c(t.A_2) = \alpha$: then N/t generates a tuple α/t in $\psi_1(t.A_1) \vee \psi_2(t.A_2)$ (indeed $\min(N, \max(\alpha, \alpha)) = \alpha$). It also generates a tuple α/t in σ_{ψ_1} and in σ_{ψ_2}. Thus, it generates a tuple α/t in $(\sigma_{\psi_1}(r) \cup \sigma_{\psi_2}(r))$.

5 Making Selection Queries More Flexible

If one assumes that the relation concerned by a selection condition is a base relation (i.e., where all the tuples have a degree $N = 1$), a tuple in the result is uncertain iff it involves at least one suspect value concerned by the selection condition. If such a tuple involves several such suspect values, it will be no more uncertain ($N = \alpha$) than

if it involves only one. However, one may find it desirable to distinguish between these situations. For instance, considering the query

$$\sigma_{job=\text{`}Engineer\text{'} \text{ and } city=\text{`}Boston\text{'} \text{ and } age=30}(Emp)$$

the tuple ⟨John, Engineer*, Boston, 30⟩ could be judged more satisfactory (less risky) than, e.g., ⟨Bill, Engineer*, Boston*, 30⟩, itself more satisfactory than ⟨Paul, Engineer*, Boston*, 30*⟩.

For a selection condition $\psi = \psi_1 \wedge \ldots \psi_m$ and a tuple t, this amounts to saying that "every attribute value (certain and suspect) of t must satisfy the condition ψ_i that concerns it, and the less there are suspect values concerned by a ψ_i in t, the more t is preferred". In other words, the selection condition becomes:

$\psi_1 \wedge \ldots \wedge \psi_m$ and *as many* $(t.A_1, \ldots, t.A_m)$ *as possible* are totally certain.

In a user-oriented language based on the algebra described above, one may then introduce an operator IS CERTAIN (meaning "is totally certain"), in the same way as there exists an operator IS NULL in SQL.

The fuzzy quantifier [12] *as many as possible* (*amap* for short) corresponds to a function from [0, 1] to [0, 1]. Its associated membership function μ_{amap} is such that: i) $\mu_{amap}(0) = 0$, ii) $\mu_{amap}(1) = 1$, iii) $\forall x, y, x > y \Rightarrow \mu_{amap}(x) > \mu_{amap}(y)$. Typically, we shall take $\mu_{amap}(x) = x$.

The selection condition as expressed above is made of two parts: a "value-based one" — that may generate uncertain answers —, and a "representation-based" one that generates gradual answers. A tuple of the result is assigned a satisfaction degree μ (seen as the complement to 1 of a suspicion degree), on top of its certainty degree N. For a conjunctive query made of m atomic conjuncts ψ_i, the degree μ associated with a tuple t is computed as follows:

$$\mu(t) = \mu_{amap}\Big(\frac{\sum_{i=1}^{m} certain(t, i)}{m}\Big) \tag{10}$$

where

$$certain(t, i) = \begin{cases} 1 \text{ if } \psi_i \text{ if of the form } A\ \theta\ q \text{ and } c(t.A) = 1, \\ 1 \text{ if } \psi_i \text{ if of the form } A_1\ \theta\ A_2 \text{ and } \min(c(t.A_1), c(t.A_2)) = 1, \\ 0 \text{ otherwise.} \end{cases}$$

In order to display the result of the query, one rank-orders the answers on N first, then on μ (in a decreasing way in both cases).

Example 7. Let us consider the relation represented in Table 1 (top) and the selection query $\sigma_\psi(Emp)$ where ψ is the condition

job = 'Engineer' and *city* = 'Boston' and *age* > 30 and
amap (*job* IS CERTAIN, *city* IS CERTAIN, *age* IS CERTAIN)

Let us assume that the membership function associated with the fuzzy quantifier *amap* is $\mu_{amap}(x) = x$. The result of the selection query is represented in Table 8 (bottom). $\diamond$

Table 8 Relation *Emp* (top), result of the selection query (bottom)

#id	*name*	*city*	*job*	*age*	*N*
38	John	Boston*	Engineer*	32	1
54	Mary	Quincy*	Engineer*	35	1
72	Bill	Boston	Engineer	40	1
81	Paul	Boston*	Engineer*	31*	1
93	Phil	Boston	Engineer	52*	1

#id	*name*	*city*	*job*	*age*	*N*	μ
72	Bill	Boston	Engineer	40	1	1
93	Phil	Boston	Engineer	52*	α	0.67
38	John	Boston*	Engineer*	32	α	0.33
81	Paul	Boston*	Engineer*	31*	α	0

This extended framework, where two degrees (N and μ) are associated with each tuple in the relations, can be easily made compositional. One just has to manage the degrees μ, in the definition of the algebraic operators, as in a gradual (fuzzy) relation context, see [7]. In base relations, it is assumed that $\mu(t) = 1\ \forall t$.

Notice that an alternative solution, of a more qualitative nature, is also possible for discriminating the tuples that (somewhat certainly) satisfy a conjunctive selection condition. It consists in using the lexicographic ordering (denoted by $>_{lex}$ hereafter) as follows. For every tuple t of the result, one builds a vector $V(t)$ of m values (1 or α):

$$V(t)[i] = \begin{cases} 1 \text{ if } \psi_i \text{ if of the form } A\ \theta\ q \text{ and } c(t.A) = 1, \\ 1 \text{ if } \psi_i \text{ if of the form } A_1\ \theta\ A_2 \text{ and } \min(c(t.A_1),\ c(t.A_2)) = 1, \\ \alpha \text{ otherwise.} \end{cases}$$

Then $V(t)$ is transformed into $V'(t)$ by ordering the i components in decreasing order (first the 1's, then the α's if any). Finally,

$$t \succ t' \Leftrightarrow V'(t) >_{lex} V'(t') \tag{11}$$

where $t \succ t'$ means that t is considered a more satisfactory answer than t'.

A refinement consists in taking into account that atomic conditions in ψ may have different importance levels. Let us denote by $w(\psi_i)$ the weight (importance level) associated with the atomic condition ψ_i and let us assume that the greater $w(\psi_i)$, the more important ψ_i. With the first method (based on scores), the idea consists in using a weighted mean instead of an arithmetic mean in Formula 10 for computing $\mu(t)$. With the second method (based on lexicographic ordering), the idea would be,

in case of ties, to use the complete preorder on the importance levels of the attributes for introducing another lexicographic refinement in order to break these ties.

6 Conclusion

In this paper, we have proposed a database model and defined the associated algebraic operators for dealing with the situation where some attribute values in a dataset are suspect, i.e., have an uncertain validity, in the absence of further information about the precise levels of uncertainty attached to such suspect values. The framework used is that of possibility theory restricted to a certainty scale made of three levels. It is likely that the idea of putting some kind of tags on suspect values/tuples/answers is as old as information systems. However, the benefit of handling such a symbolic tag in the framework of possibility theory is to provide a rigorous setting for this processing.

A crucial point is that the data complexity of all of the algebraic operations is the same as in the classical database case, i.e., it is either linear or polynomial, which makes the approach perfectly tractable. Moreover, the definitions of both the model and the operators are quite simple and do not raise any serious implementation issues.

It is worth mentioning that this model could be rather straightforwardly extended in order to handle disjunctive suspect values — of the form $(v_1 \vee \ldots v_n)^*$ — instead of singletons. The definitions of the operators would then become a bit more complicated, but the data complexity would not change. Another extension would be to use a small set of suspicion levels rather than one. This would bring us closer to the general setting of the certainty-based model described in [9].

References

1. Bosc, P., Pivert, O.: About projection-selection-join queries addressed to possibilistic relational databases. IEEE Trans. on Fuzzy Systems **13**(1), 124–139 (2005)
2. Bosc, P., Pivert, O., Prade, H.: A model based on possibilistic certainty levels for incomplete databases. In: Godo, L., Pugliese, A. (eds.) SUM 2009. LNCS, vol. 5785, pp. 80–94. Springer, Heidelberg (2009)
3. Dubois, D., Prade, H.: Possibility Theory. Plenum, New York (1988)
4. Haas, P.J., Suciu, D.: Special issue on uncertain and probabilistic databases. VLDB J. **18**(5), 987–988 (2009)
5. Hastings, D., McManus, H.: A framework for understanding uncertainty and its mitigation and exploitation in complex systems. In: Proc. of Engineering Systems Symposium MIT (2004)
6. Imielinski, T., Lipski, W.: Incomplete information in relational databases. J. of the ACM **31**(4), 761–791 (1984)
7. Pivert, O., Bosc, P.: Fuzzy Preference Queries to Relational Databases. Imperial College Press, London (2012)
8. Pivert, O., Prade, H.: Database querying in the presence of suspect values. In: Morzy, T., Valduriez, P., Bellatreche, L. (eds.) ADBIS 2015. CCIS, vol. 539, pp. 44–51. Springer, Heidelberg (2015)

9. Pivert, O., Prade, H.: A certainty-based model for uncertain databases. IEEE Transactions on Fuzzy Systems (2015) (to appear)
10. Suciu, D., Olteanu, D., Ré, C., Koch, C.: Probabilistic Databases. Synthesis Lectures on Data Management. Morgan & Claypool Publishers (2011)
11. Zadeh, L.: Fuzzy sets as a basis for a theory of possibility. Fuzzy Sets and Systems **1**(1), 3–28 (1978)
12. Zadeh, L.: A computational approach to fuzzy quantifiers in natural languages. Computing and Mathematics with Applications **9**, 149–183 (1983)