An Extended Relational Database Model For Uncertain And Imprecise Information

Suk Kyoon Lee

Department of Computer Science,
University of Iowa,
Iowa City, IA. 52242
sklee@herky.cs.uiowa.edu

Abstract

We propose an extended relational database model which can model both uncertainty and imprecision in data. This model is based on Dempster-Shafer theory which has become popular in AI as an uncertainty reasoning tool. The definitions of *Bel* and *Pls* functions in Dempster-Shafer theory are extended to compute the beliefs of various comparisons (e.g., equality, less than, etc.) between two basic probability assignments. Based on these new definitions of *Bel* and *Pls* functions and the Boolean combinations of *Bel* and *Pls* values for two events, five relational operators such as Select, Cartesian Product, Join, Projection Intersect, and Union are defined.

Keywords: *database systems, uncertainty, Dempster-Shafer theory, Artificial Intelligence, relational algebra, uncertain and imprecise information.*

1. Introduction

In the real world, data is often imperfect, not just because of the unreliability of its source, but also because of its nature. Consider weather reports, doctor's medical diagnoses, or any data that has stochastic features. One of our goals in computer science is to design intelligent software which can store imperfect information and interpret queries about it. Most data in biology, genetics, and physics are stochastic in nature. When database systems are designed for these fields, they should be able to handle imperfect information.

In [Duboi86], *imprecision* and *uncertainty* are stated as two complementary aspects of imperfect information, since imprecision refers to the contents of data and uncertainty refers to the degree of truth of data. Imprecision in data is usually modeled by exclusive disjunction as in [Morri90], [Lipsk79], [Willi88], [Ola 92]. For example, when the location of some conference for the next year, which should be either in Europe or U.S.A., has not been determined, we represent this information as {Europe, U.S.A.}. Several ways to represent uncertainty in data exist: the approaches based on fuzzy set and possibility theory [Zadeh78] and the approaches based on probability theory. Refer to [Lee 92] for the references of these approaches. Though approaches based on fuzzy set and possibility theory seem to be good solutions for problems that arise due to lexical imprecision, it is difficult to justify these approaches for the areas where stochastic models are very popular. There are relatively few works based on probability theory.

Though there are many cases where modeling both imprecision and uncertainty in data is quite helpful, few works have looked into modeling both imprecision and uncertainty in database systems. We proposed a new database model which can both model imprecision and uncertainty in data using Dempster-Shafer theory of belief [Lee 92]. We often need to represent imprecision in the probability distribution of a data object for the same reason that we need to represent imprecision in a certain data value just like the conference example given above. Suppose we know more information about the conference location: We have the following probability distribution: Europe/0.5 + U.S.A./0.5. If it is in Europe, it will be in Paris or Vienna. If it is in U.S.A., it will be in Phoenix, Iowa City, or Kansas City. We don't know any probability distribution for these locations. In this case, it is natural to represent this information as {Paris, Vienna}/0.5 + {Phoenix, Iowa City, Kansas City}/0.5. Note that the imprecision in this example is due to the lack of information of probability distribution between the cities. Representing this kind of lack of information is one of motivations behind Dempster-Shafer theory. Modeling both imprecision and uncertainty in data

Proceedings of the 18th VLDB Conference
Vancouver, British Columbia, Canada 1992

makes sense in the cases where the nature of information is uncertain and the complete information is often not available.

Dempster-Shafer theory [Shafe76], which has attracted much attention in AI as a tool to handle uncertain information, has become a standard tool in expert systems applications [Duboi87], [Abel 88], [Falke88], [Li 88], [Prova90]. In [Lee 92], we proposed an extended database model based on Dempster-Shafer theory, but we did not define the Select operation which involves the comparisons between two attributes as well as all the other relational operators. In this paper, we propose a solution to these problems. This paper is organized as follows: in Section 2, we briefly review Dempster-Shafer theory and extend it so that the belief of the various comparisons of two basic probability assignments can be computed. We review the extended relational model proposed in [Lee 92] in Section 3. In Section 4, we define five relational operators (Select, Join, Cartesian Product, Intersect, Union) and give examples. In Section 5, we summarize the paper and discuss our future work.

2. Dempster-Shafer Theory

2.1. Background

A *universe*, or *frame of discernment* is a set of mutually exclusive and exhaustive hypotheses about some problem domain [Shafe76]. A body of evidence describing some uncertain information can be represented in the following way:

Definition 2.1: Let D be a frame of discernment, then, a function m: $2^D \rightarrow [0,1]$ is called a *basic probability assignment* whenever

(a) $m(\emptyset) = 0$. (b) $\sum_{A \subseteq D} m(A) = 1$.

and, an element A of 2^D is called a *focal element* whenever $m(A) > 0$.

$m(A)$ shows a relative confidence exactly in A, not in any subset of A. Since m can be viewed as a probability measure on 2^D (not on D), each focal element need not be disjoint, nor form a covering of D. In fact, D can be a focal element; $m(D)$ is interpreted as the level of confidence for ignorance. For instance, $m(D) = 1$ means total ignorance. The most two important functions based on a basic probability assignment m are the *belief* and *plausibility* functions given as follows:

Definition 2.2: (*Bel* and *Pls* functions) For some given m, the belief function *Bel* and plausibility function *Pls* for an event $B \subset D$, in the sense of probability theory, are defined as:

(a) $Bel(B) = \sum_{A \subseteq B} m(A)$.

(b) $Pls(B) = \sum_{A \cap B \neq \emptyset} m(A) = 1 - Bel(\neg B)$.

and, if all focal elements of *Bel* are singletons, then *Bel* is called *Bayesian*.

$Bel(B)$ reflects the total weight of evidence (belief) in B, while $Pls(B)$ reflects the total weight of evidence which is not committed to $\neg B$ where $\neg B = D - B$, (complement of B). For more information about Dempster-Shafer theory, refer to [Shafe76], [Shafe86], [Shafe87], [Halpe90], [Orpon90].

2.2. Extended Definitions of *Bel* and *Pls* functions

In conventional probability theory, we can easily compute the probability of the comparison of two independent probability distributions. But, Dempster-Shafer theory does not have the corresponding definition of *Bel* and *Pls* functions to handle the comparison of two independent basic probability assignments. When we use basic probability assignments to represent uncertain and imprecise information instead of probability distributions, it is necessary to extend the definition of *Bel* and *Pls* function which can handle the comparison of two independent basic probability assignments. For example, let's suppose that we have uncertain and imprecise information about the blood types of Jim and his wife Kim. Now we want to compute the *Bel* and *Pls* values of the case that their blood types are equal. The current Dempster-Shafer theory cannot compute the *Bel* and *Pls* values of such events.

Let's think about the probability of the various comparisons of two probability distributions. Let X and Y be two random variables which are independent and whose probability functions are P_X, P_Y: $D \rightarrow [0, 1]$. Then

$$Pr(X = Y) = \sum_{a \in D} P_X(a) * P_Y(a)$$

$$Pr(X < Y) = \sum_{a \in D} P_X(a) * \sum_{b \in D \wedge a < b} P_Y(b)$$

Since a basic probability assignment is a probability distribution of the power set of a domain, this idea can be used to compute the *Bel* and *Pls* values of the various comparisons of the probability distributions on the power set of a domain. Then m_X, m_Y: $2^D \rightarrow [0, 1]$ will be used instead of P_X, P_Y.

Definition 2.3: Let X, Y be two random variables which are independent and whose probability functions (basic probability assignments) are m_X, m_Y: $2^D \rightarrow [0, 1]$, respectively. Then, for $a, b \subset D$,

$$\begin{aligned} Bel(X = Y) &= \sum_{|a| = 1} m_X(a) * m_Y(a). \\ Pls(X = Y) &= \sum_{a \cap b \neq \varnothing} m_X(a) * m_Y(b) \\ &= 1 - \sum_{a \cap b = \varnothing} m_X(a) * m_Y(b). \end{aligned}$$

The definition of $Bel(X = Y)$ is quite intuitive[1]. Suppose $f(a, b) = m_X(a) * m_Y(b)$ be a joint basic probability assignment between two independent random variables X, Y in the same way as a joint probability distribution between two independent random variables is defined in probability theory. Then, $\{<a, a> \mid a \in D\}$ is used for the event $(X = Y)$. $Bel(X = Y)$ is the sum of the multiplication of the degrees of support from two different sources represented by random variables X and Y for every singleton element a in a domain D, while $Pls(X = Y)$ represents 1 – the sum of the multiplication of the degrees of support from two sources for any two subsets a, b when they have no common elements. For example, suppose that we have the following information about the blood types for Jim and Kim. $BloodType_{Jim}$ = {A}/0.5 + {A, B}/0.5 and $BloodType_{Kim}$ = {A}/0.3 + {B}/0.3 + {A, B}/0.4. Then, $Bel(X = Y) = 0.15$ and $Pls(X = Y) = 0.15 + 0.20 + 0.15 + 0.15 + 0.20 = 0.75$ where X and Y denote the random variables for the blood types for Jim and Kim. Note that when $BloodType_{Jim}$ = {A, B}/1.0 and $BloodType_{Kim}$ = {A, B}/1.0, $Bel(X = Y) = 0$ instead of 1. From the definition above, the following result can be easily derived.

Theorem 2.1: For two independent random variables X, Y whose probability functions (basic probability assignments) are m_X, m_Y : $2^D \rightarrow [0, 1]$, respectively,

(1) $Bel(X \neq Y) = \sum_{a \cap b = \varnothing} m_X(a) * m_Y(b)$, where $a,b \subset D$

(2) $Pls(X \neq Y) = \sum_{a \subset D} m_X(a) * \sum_{b \subset D \wedge [a = b \rightarrow |a| > 1]} m_Y(b)$

(Proof) From Definition 2.2 (b) and Definition 2.3, (1) and (2) can be directly derived.■

For the non-equality comparisons, we can have the similar definition and theorem. Next, the definition for $Bel(X < Y)$ and $Pls(X < Y)$ is introduced.

Definition 2.4: Let X, Y be two random variables which are independent and whose probability functions (basic probability assignments) are m_X, m_Y: $2^D \rightarrow [0, 1]$, respectively. Then,

$$Bel(X < Y) = \sum_{a \subset D} m_X(a) * \sum_{b \subset D \wedge a <^{\forall} b} m_Y(b)$$
$$\text{where } a <^{\forall} b \equiv \forall c \in b\ [a < c]$$

$$Pls(X < Y) = \sum_{a \subset D} m_X(a) * \sum_{b \subset D \wedge a <^{\exists} b} m_Y(b)$$
$$\text{where } a <^{\exists} b \equiv \exists c \in b\ [a < c]$$

The computation of $Bel(X < Y)$ and $Pls(X < Y)$ is based on the comparison between a focal element of m_X and a focal element of m_Y. When every member of a focal element of m_X is less than every member of a focal element of m_Y, the multiplication of the corresponding probabilities is included in the computation of $Bel(X < Y)$. Similarly, when every member of a focal element of m_X is less than some member of a focal element of m_Y, the multiplication of the corresponding probabilities is included in the computation of $Pls(X < Y)$. We have the following theorem.

Theorem 2.2 : For two independent random variables X, Y whose probability functions (basic probability assignments) are m_X, m_Y : $2^D \rightarrow [0, 1]$, respectively,

(1) $Bel(X \leq Y) = \sum_{a \subset D} m_X(a) * \sum_{b \subset D \wedge a \leq^{\forall} b} m_Y(b)$
where $a \leq^{\forall} b \equiv \forall c \in b\ [a \leq c]$

(2) $Pls(X \leq Y) = \sum_{a \subset D} m_X(a) * \sum_{b \subset D \wedge a \leq^{\exists} b} m_Y(b)$
where $a \leq^{\exists} b \equiv \exists c \in b\ [a \leq c]$

(Proof) From Definition 2.2 (b) and Definition 2.4, (1) and (2) can be directly derived.■

Note that $Bel(X \leq Y) \geq Bel(X < Y) + Bel(X = Y)$. For example, we have the following information about the college GPAs of Jim and Kim. GPA_{Jim} = {C}/0.3 + {B, C}/0.7 and GPA_{Kim} = {B}/1.0. Let X and Y denote the random variables for the GPAs of Jim and Kim. Then, $Bel(X \leq Y) = 1$, but $Bel(X < Y) = 0.3$ and $Bel(X = Y) = 0$. This relation also applies to other inequality comparison cases and the Pls function, too. In addition to the previous two theorems, we can easily prove that the definitions of $Bel(X\ \theta\ Y)$ and $Pls(X\ \theta\ Y)$ where $\theta \in \{=, \neq, <, \leq\}$ satisfy the following properties.

(1) $0 \leq Bel(X\ \theta\ Y) \leq Pls(X\ \theta\ Y) \leq 1$.
(2) $Bel(X\ \theta\ Y) + Bel(\neg(X\ \theta\ Y)) \leq 1$.
(3) $Pls(X\ \theta\ Y) + Pls(\neg(X\ \theta\ Y)) \geq 1$.
(4) If all the focal elements of m_X, m_Y are singletons,

[1] To the best of our knowledge, we don't know any similar definition of $Bel(X = Y)$ and $Pls(X = Y)$ appearing in the literature. Hence, we could not compare this definition with others.

then $Bel(X\ \theta\ Y)$ is *Bayesian* (i.e., $Bel(X\ \theta\ Y) = Pls(X\ \theta\ Y) = Pr(X\ \theta\ Y)$).

From now on, we introduce a random variable in the notation of *Bel* and *Pls* functions so that we can represent multiple basic probability assignments and their *Bel* and *Pls* values. For example, when we have two basic probability assignments m_X and m_Y, let $Bel(c \mid X)$, $Pls(c \mid X)$ and $Bel(c \mid Y)$, $Pls(c \mid Y)$ represent $Bel(c)$, $Pls(c)$ based on m_X and $Bel(c)$, $Pls(c)$ based on m_Y, respectively.

3. Extended Database Model

3.1. Data Representation

A *domain* (frame of discernment) is a finite set of mutually exclusive and exhaustive values. Let t be a data object (*tuple*), a_i be an attribute of t, and D_j be a domain of a_i. An attribute a_i is a mapping from a set of data objects to a domain $D_j \cup \{\perp\}$ where $\perp$ represents an undefined value, and $t.a_i$ represents the mapped value in a domain $D_j \cup \{\perp\}$. The inclusion of $\perp$ in the range of a_i allows us to handle the special case where applying an attribute a_i to an data object does not make sense.

One of the major features of the conventional relational database model is that every attribute value is atomic [Date 86]. In order to represent imprecise and uncertain information, we should modify this feature. As an attribute value, a set of values should be allowed for the representation of imprecise data, while a probability distribution should be allowed for the representation of uncertain data.

Definition 3.1: For any data object t_i and its any relevant attribute a_j, let D_k denote the domain the attribute maps into, and m_{ij} represent the basic probability assignment for a data object t_i and an attribute a_j. Then, the attribute value $t_i.a_j = \{ < d, m_{ij}(d) > \mid d \in D_k \cup \{\perp\} \wedge m_{ij}(d) > 0 \}$.

This definition says that a probability distribution of the power set of a domain is allowed in every attribute value. As an example, we use information about Mary's health record t_m.

- (t_m).disease = $< \{ d_1, d_3 \}, 0.6 >, < d_2, 0.4 >$[2]. This attribute value explains that we believe the disease Mary has is either d_2 with a probability 0.4, or one of $\{ d_1, d_3 \}$ with a probability 0.6. But, we don't know how probable each element of $\{ d_1, d_3 \}$ is, but one of them is sure with a probability 0.6.

The traditional null value can be naturally handled using a set. The null value is subdivided into three different cases such as *unknown*, *inapplicable*, and *unknown or inapplicable*, denoted by the special strings *nk*, *na*, and *nka*, respectively. The string *nk* represents the corresponding domain *D* itself for an attribute. Similarly, *na*, and *nka* represent $\{\perp\}$ and $D \cup \{\perp\}$, respectively. For example,

- (t_m).eye_color = < *nk*, 1.0 >. When the corresponding domain is { black, brown, blue }, this attribute value means that her eye is in one of colors { black, brown, blue }, but we don't know which one it is. Put it another way, every value in the domain is possible.
- (t_m).husband_name = < *na*, 1.0 >. This means that she does not have any husband.
- (t_m).husband_name = < *nka*, 1.0 >. This means we don't know even whether she has a husband at all.

The above three examples demonstrate how to represent and interpret null values in our model. Refer to [Lee 92] for detailed explanation and more complicate examples.

3.2. Extended Relational Database Model

In the conventional relational database, information is represented by the set-theoretic *relation*, which is a subset of the Cartesian product of a list of domains $D_1 \times D_2 \times \ldots D_n$. With the data representation for each attribute value of a data object introduced in the previous section, which is a probability distribution on the power set (basic probability assignment) of a domain instead of an atomic value, the definition of a *relation* is changed to the following way.

Definition 3.2: A *relation* (or *table*) *T* based on $D_1, D_2, \ldots D_n$ is defined as $T \subset G_1 \times G_2 \times \ldots \times G_n \times CL$ where G_i is a set of all the probability distributions on the power set of a domain D_i and $CL = \{< b, p > \mid b, p \in [0, 1] \wedge b \leq p\}$

Each G_i corresponds to a domain whose element can be interpreted as a set of pairs of a focal element and its basic probability for some basic probability assignment *m*. In the set *CL*[3], a pair of values $< b, p >$ represents the confidence level in every tuple in a relation *T*. Specifically, *b* represents the belief value for the corresponding tuple, while *p* is the plausibility value.

A *relation* can be viewed as a *table* with rows and columns, where each column corresponds to an *attribute* and each row corresponds to a *tuple* which represents a data object. Let a *heading* represent a set of attributes and a *body* be a set of tuples in a table. With a tuple *t*, let $A(t)$ represent the finite, non empty set of attributes

[2] The exact notation is $\{ < \{ d_1, d_3 \}, 0.6 >, < \{ d_2 \}, 0.4 > \}$. But, without causing ambiguity, the set notation is omitted.

[3] *CL* will be used also as a system attribute name which every table has.

NAME	STATUS	GRADE IN FINANCE	GRADE IN ART	GPA	RECOMME.	*CL*
Tom	{SE, GR}	A	<A,0.5>, <*nk*,0.5>	{2.5-3}	{3,4}	[1 ; 1]
David	FR	{B,C}	*na*	{2.7-3.0}	{4,5}	[0.7 ; 1]
Bob	{SO, JU}	<{B,C},0.5>,<B,0.5>	<{B,C},0.6>,<C,0.4>	<{2.7-3},0.5 >, <{2.5-3.3},0.5>	5	[1 ; 1]
Jane	<GR, 0.7>, <SE, 0.3>	<A,0.3>,<B,0.4>,<C,0.3>	<A,0.2>,<B, 0.3>, <C,0.5>	2.5	2	[0.7; 0.7]
Jill	*nk*	A	{A,B}	3.1	1	[0.2; 0.8]

Figure 1: Job-Applicants $T_{Applicants}$

relevant to the tuple t not including *CL*. Also, with a table T, let $A(T)$ represent the set of attributes of T except *CL* and $D(T)$ be a function which associates each attribute in $A(T)$ with the corresponding domain.

With this definition of a relation (table), an imprecise and uncertain attribute value can be stored in each cell of a table. Let's suppose that we have a table $T_{Applicants}$ in Figure 1 which contains information about part-time job applicants to some company[4]. The attribute "STATUS" indicates the status of an applicant in a college. The attribute "RECOMME." shows how strongly each applicant is recommended, with 5 for the highest and 1 for the lowest. Note that some of the values in the column *CL* are values other than [1 ; 1], since we don't have full belief in these tuples. A probability value in a cell is omitted when it is 1, for the notational simplicity. The status of Tom is either senior or graduate, and his grade in finance is *A*. His grade in art is known to be *A* with probability value 0.5. As you see in this example, we introduced two levels of uncertainty. One is for an attribute value, while the other is for a tuple. In most cases, the uncertainty value of a tuple (*CL*) comes from the credibility of the source. In the next section, we will discuss about how to evaluate queries in this database model.

4. Generalized Relational Algebra

4.1. Select

4.1.1. *principle*

Generally, a select operation extracts from a table the tuples whose specified attributes values satisfy a given condition, and returns them as a new table. A select operation when applied to a table T with a condition B is denoted by T ***where*** (B). A condition B can be either an atomic condition, or a compound condition constructed from an atomic condition by logical connectives (conjunction, disjunction, negation). An atomic condition B is a simple comparison which has the form $a\ \theta\ a'$ or $a\ \theta\ c$ where $a \in A(T)$, $a' \in A(T)$, $(D(T))(a) = (D(T))(a')$, $c \subset (D(T))(a)$[5] and θ is a simple comparison operator $=$, $\neq$, $<$, $>$, $\leq$, $\geq$. In this paper, we will show how to handle simple queries of the form $a\ \theta\ a'$ as well as simple queries of the form $a\ \theta\ c$ and compound queries.

Since a probability distribution on the power set of a domain (basic probability assignment) is used for data representation, it is natural to use *Bel* and *Pls* functions to evaluate how a tuple satisfies a given condition. Informally, T ***where*** (B) returns a table which has the same contents of the table T except in the *CL* column. The values of *CL* (confidence level) in the resulting table reflect the support level of each tuple for the given condition B. In order to give a formal definition of a select operation in our model, first we need to define one auxiliary function which is about the Boolean combination of *Bel* and *Pls* values. Then, we will try to formalize the selection operation of the form $a\ \theta\ c$, $a\ \theta\ a'$, and compound queries in sequence. First of all, let's define a function which computes the pair of *Bel* and *Pls* values of the conjunction and disjunction of two events.

Definition 4.1: For independent events E_1 and E_2 where $Bel(E_1) = b_1$, $Pls(E_1) = p_1$, and $Bel(E_2) = b_2$, $Pls(E_2) = p_2$, the pairs of *Bel* and *Pls* values of the conjunction and disjunction of the events E_1 and E_2 are defined by a function δ as following:

$$[Bel(E_1 \wedge E_2) \,;\, Pls(E_1 \wedge E_2)] = \delta([b_1 \,;\, p_1], [b_2 \,;\, p_2], 0, \wedge) = [b_1 * b_2 \,;\, p_1 * p_2]$$

$$[Bel(E_1 \vee E_2) \,;\, Pls(E_1 \vee E_2)] = \delta([b_1 \,;\, p_1], [b_2 \,;\, p_2], 0, \vee) = [1 - (1 - b_1) * (1 - b_2) \,;\, 1 - (1 - p_1) *(1 - p_2)]$$

Let's call the resulting pair from $\delta([b_1 \,;\, p_1], [b_2 \,;\, p_2], 0, \wedge)$ as the *conjunctive combination* of $[b_1 \,;\, p_1]$ and $[b_2 \,;\, p_2]$. The *disjunctive combination* is defined in the similar way. Note that the assumption about independent events causes the third parameter in the

[4] This example is not very realistic, but just for the purpose of exposition of the concepts. A similar example is found in [Lee 92].

[5] Note that imprecise queries are allowed in the select operation.

NAME	STATUS	GRADE IN FINANCE	GRADE IN ART	GPA	RECO-MME.	*CL*
Tom	{SE, GR}	A	<A,0.5>, <*nk*,0.5>	{2.5-3}	{3,4}	[0 ; 1]
Jane	<GR,0.7>, <SE, 0.3>	<A,0.3>,<B,0.4>,<C,0.3>	<A,0.2>,<B,0.3>,<C,0.5>	2.5	2	[0.49; 0.49]
Jill	*nk*	A	{A,B}	3.1	1	[0;0.8]

Figure 2: $T_{Applicants}$ ***where*** (STATUS = {GR})

NAME	STATUS	GRADE IN FINANCE	GRADE IN ART	GPA	RECOMME.	*CL*
Tom	{SE, GR}	A	<A,0.5>, <*nk*,0.5>	{2.5-3}	{3,4}	[0.5 ; 1]
Bob	{SO, JU}	<{B,C},0.5>, <B,0.5>	<{B,C},0.6>, <C,0.4>	<{2.7-3},0.5>, <{2.5-3.3},0.5>	5	[0 ; 0.8]
Jane	<GR,0.7>, <SE, 0.3>	<A,0.3>,<B, 0.4>,<C,0.3>	<A,0.2>,<B, 0.3>, <C,0.5>	2.5	2	[0.23; 0.23]
Jill	*nk*	A	{A,B}	3.1	1	[0; 0.8]

Figure 3: $T_{Applicants}$ ***where*** (GRADE IN FINANCE = GRADE IN ART)

NAME	STATUS	GRADE IN FINANCE	GRADE IN ART	GPA	RECOMME.	*CL*
Tom	{SE, GR}	A	<A,0.5>,<*nk*,0.5>	{2.5-3}	{3,4}	[1 ; 1]
David	FR	{B,C}	*na*	{2.7-3.0}	{4,5]	[0.7 ; 1]
Bob	{SO, JU}	<{B,C},0.5>, <B,0.5>	<{B,C},0.6>, <C,0.4>	<{2.7-3},0.5>, <{2.5-3.3},0.5>	5	[0.5 ; 1]

Figure 4: $T_{Applicants}$ ***where*** ((RECOMME. = {3, 4, 5}) ∧ (GPA = {2.5 - 3.1}))

function δ to be zero. The more general form of the function δ introduced in [Lee 92] is $\delta([b_1 ; p_1], [b_2 ; p_2], \rho, \wedge)$ and $\delta([b_1 ; p_1], [b_2 ; p_2], \rho, \vee)$ where ρ is a dependent factor (*correlation* in the statistical terms) between two events E_1 and E_2. Here, the independence assumption is just for the purpose of exposition. Whenever information about the dependency factor is available, we can use the general form at any time.

As a table is a set of tuples, before we can give a definition of a select operation, we need to define some function which is working on the tuple level with a given condition. Let's consider the case where a condition is atomic and of the form $a \ \theta \ c$.

Definition 4.2: For any tuple t in a table T, and a condition B which is a simple equality comparison between a and c where $a \in A(T)$, $c \subset (D(T))(a)$, the function φ which takes a tuple t and a condition B and returns a pair of *Bel* and *Pls* values is defined as

$$\varphi(t, a = c) = \delta(t.CL, [\, Bel(c \mid X) \, ; Pls(c \mid X) \,], 0, \wedge)$$

where X is a random variable for the basic probability assignment which $t.a$ is based on.

The resulting pair of *Bel* and *Pls* values is the conjunctive combination of the *CL* value of a tuple t and the pair of *Bel* and *Pls* values of an event c based on the basic probability assignment which the attribute value $t.a$ represents. We assume that the independence relation holds in this definition. This assumption is reasonable, since we distinguish between an attribute level uncertainty and a tuple level uncertainty which usually reflects the degree of confidence of the information source. Note that this definition is different from the one in [Lee 92] where we implicitly assumed that every *CL* value in a base table is [1 ; 1]. We dropped that assumption in this paper. Here, we only show the equality comparison case. Extending the definition to the other non equality comparison cases is straightforward by converting the non equality comparison in terms of the equality comparison. For more information on the other comparison cases such as ≠, <, >, ≤, ≥, refer to [Lee 92]. Now, we are ready to show how to handle a tuple with an atomic condition of the form $a \ \theta \ a'$.

Definition 4.3: For any tuple t in a table T, and a condition B which is the simple comparison between a_i and a_j where $a_i \in A(T)$, $a_j \in A(T)$, $(D(T))(a_i) = (D(T))(a_j)$, the function φ which takes a tuple t and a condition B and returns a pair of *Bel* and *Pls* values is defined as

$$\varphi(t, a_i \ \theta \ a_j) = \delta(t.CL, [Bel(X_i \ \theta \ X_j), Pls(X_i \ \theta \ X_j)], 0, \wedge)$$

where θ is a simple comparison operator and X_i, X_j are the random variables for the basic probability assignments which $t.a_i$, $t.a_j$ are based on, respectively.

This definition is based on the conjunctive combination of two pairs of *Bel* and *Pls* values, and the independence assumption.

Now, we show how to handle a compound condition consisting of logical connectives (negation, conjunction, disjunction).

Definition 4.4: For any tuple t in a table T, and conditions B_1, B_2 which are two events with a dependency factor ρ, and the compound comparisons between a_i and a_j, or between a_i and c_k where $a_i \in A(T)$, $a_j \in A(T)$, $(D(T))(a_i) = (D(T))(a_j)$, $c_k \subset (D(T))(a_i)$, then function φ is defined as

$$\varphi(t, \neg B_1) = [1 - pls;\ 1 - bel], \text{ where } [bel; pls] = \varphi(t, B_1)$$
$$\varphi(t, (B_1 \wedge B_2)) = \delta(\varphi(t, (B_1)), \varphi(t, (B_2)), \rho, \wedge)$$
$$\varphi(t, (B_1 \vee B_2)) = \delta(\varphi(t, (B_1)), \varphi(t, (B_2)), \rho, \vee).$$

Note that we dropped the independence assumption between two events in the function δ. Think about the following compound event : disease = $\{d_1\}$ $\vee$ disease $\neq$ $\{d_1\}$. The independence assumption will underestimate the resulting CL value in this example. If the information about ρ is available, the function δ will adjust the CL value properly. [Lee 92] shows how to adjust the CL value with ρ and how to handle the case that the information about ρ is not available. Before giving the definition of a select operation, let's define one more auxiliary function.

Definition 4.5: For any tuple t and a pair of Bel and Pls values $[bel\ ;\ pls]$, the function r which takes t and $[bel\ ;\ pls]$ returns a new tuple t' where

$$(\forall b)_{b \in A(T)}\ [\ t.b = t'.b \wedge t'.CL = [bel\ ;\ pls]\ ].$$

In other words, the function r replaces the CL value in a tuple t by a new CL value. Now, we are ready to give a formal definition of a select operation.

Definition 4.6: For a table T and a simple or compound condition B, a select operation T ***where*** (B) is defined as

$$T \textbf{ where } (B) = \{\ r(t, \varphi(t, B)) \mid t \in T \wedge P_{threshold}(\varphi(t, B))\}$$

where $P_{threshold}$ is the system predicate defining the threshold values.

The purpose of the predicate $P_{threshold}$ is filtering out tuples with very low CL values, since we don't want to keep every possible tuples. For example, we don't want to keep tuples with [0 ; 0] as a CL value. In this paper, we will set the Pls value to be greater than 0 as a threshold value for the purpose of examples. Formally, $P_{threshold}([b\ ;\ p]) = p > 0$.

4.1.2. *examples*

Based on the table $T_{Applicants}$ in Figure 1, suppose we have the following queries:

A_1	A_2	A_3	CL
u_1	u_2	u_3	c_u
v_1	v_2	v_3	c_v

Figure 5 : Table T_1

A_3	A_4	CL
x_3	x_4	c_x
y_3	y_4	c_y

Figure 6 : Table T_2

A_1	A_2	A_3	A_3	A_4	CL
u_1	u_2	u_3	x_3	x_4	$c_1\ (= \delta(c_u, c_x, 0, \wedge)\ )$
u_1	u_2	u_3	y_3	y_4	$c_2\ (= \delta(c_u, c_y, 0, \wedge)\ )$
v_1	v_2	v_3	x_3	x_4	$c_3\ (= \delta(c_v, c_x, 0, \wedge)\ )$
v_1	v_2	v_3	y_3	y_4	$c_4\ (= \delta(c_v, c_y, 0, \wedge)\ )$

Figure 7 : T_1 ***times*** T_2

- Find all the applicants whose status is graduate. In our language, this query is formulated as $T_{Applicants}$ ***where*** (STATUS = {GR}). The resulting table is in Figure 2.
- Find all the applicants whose grade in finance is equal to the grade in art. This query is represented as $T_{Applicants}$ ***where*** (GRADE IN FINANCE = GRADE IN ART). The result is shown in Figure 3.
- Find all the applicants whose level of recommendation is in {3, 4, 5} and whose GPA is between 2.5 and 3.1. The result of this query $T_{Applicants}$ ***where*** ((RECOMME. = {3, 4, 5}) $\wedge$ (GPA = {2.5 - 3.1})) is shown in Figure 4.

4.2. Cartesian Product, Join, and Intersect

Let T_1, T_2 are ordinary relations on the Cartesian product of the sets of the probability distributions on the power sets of domains of the respective attributes. Then, the Cartesian product of two tables T_1, T_2 is defined in the usual way.

Definition 4.7: For any two tables T_1, T_2 the Cartesian product of T_1, T_2 is defined as

$$\begin{aligned} T_1 \textbf{ times } T_2 = \{\ t \mid A(t) = A(T_1) \cup A(T_2) \\ \wedge (\exists t_1)_{t1 \in T1} (\forall a)_{a \in A(T1)} [\ t.a = t_1.a \\ \wedge (\exists t_2)_{t2 \in T2} (\forall a)_{a \in A(T2)} [\ t.a = t_2.a \\ \wedge t.CL = \delta(t_1.CL, t_2.CL, 0, \wedge)]]\}. \end{aligned}$$

Consider the following two tables T_1 and T_2 from Figure 5 and Figure 6. Then, their Cartesian product is given by T_1 ***times*** T_2 in Figure 7. The Cartesian product is carried out in the usual way except the column CL. The value of CL in a tuple of the resulting table is the conjunctive combination of the CL values from the corresponding tuples in the table T_1 and T_2.

Next, we define the join of two tables indirectly through the Cartesian operation and select operation.

Definition 4.8: For any two tables T_1 and T_2, let $\{A_1, A_2, \ldots, A_k\} = A(T_1) \cap A(T_2)$, and $A_i^{(Tm)}$ be A_i in T_m where $1 \leq$

A_1	A_2	A_3	A_3	A_4	CL
u_1	u_2	u_3	x_3	x_4	$\delta(c_1,[Bel(X_{u3}=X_{x3}),Pls(X_{u3}=X_{x3})],0,\wedge)$
u_1	u_2	u_3	y_3	y_4	$\delta(c_2,[Bel(X_{u3}=X_{y3}),Pls(X_{u3}=X_{y3})],0,\wedge)$
v_1	v_2	v_3	x_3	x_4	$\delta(c_3,[Bel(X_{v3}=X_{x3}),Pls(X_{v3}=X_{x3})],0,\wedge)$
v_1	v_2	v_3	y_3	y_4	$\delta(c_4,[Bel(X_{v3}=X_{y3}),Pls(X_{v3}=X_{y3})],0,\wedge)$

Figure 8 : T_1 ***join*** T_2

$i \leq k$ and $1 \leq m \leq 2$. Then, the equi-join of T_1 and T_2 is defined as

$$T_1 \textbf{\textit{ join }} T_2 = (T_1 \textbf{\textit{ times }} T_2)$$
$$\textbf{\textit{where }} (A_1^{(T1)} = A_1^{(T2)} \wedge \ldots \wedge A_k^{(T1)} = A_k^{(T2)}).$$

The table T_1 ***join*** T_2 in Figure 8 is an example of the join operation based on the definition above where X_m in the CL column represents the random variable that an attribute value m is based on in the table. Note that the computation of attribute CL is the conjunctive combination of the CL values of the Cartesian product operation T_1 ***times*** T_2 from Figure 7 and the CL values from the select operation.

Next, we define the intersect operation similar to the regular intersect operation. The resulting table only consists of the common tuples belonging to both input tables, but the CL values of the resulting table is the conjunctive combination of the CL values of the corresponding tuples in both tables.

Definition 4.9: For any two tables T_1 and T_2 where $A(T_1) = A(T_2)$, the intersect of T_1 and T_2 is defined as

$$T_1 \textbf{\textit{ intersect }} T_2 = \{\, t \mid A(t) = A(T_1)$$
$$\wedge (\exists t_1)_{t1 \in T1} (\exists t_2)_{t2 \in T2} (\forall a)_{a \in A(T1)} [\, t_1.a = t_2.a$$
$$\wedge\, t.a = t_2.a \wedge t.CL = \delta(t_1.CL, t_2.CL, 0, \wedge)]\}.$$

Let table T_3 be defined as in Figure 9, then T_2 ***intersect*** T_3 is represented as in Figure 10.

4.3. Projection and Union - Redundancy

If two tuples have the same values in every attribute except the attribute CL, these tuples are called redundant. Since the projection operation throws away some attributes in a table, some tuples in the resulting table might be redundant. When the operation union is applied to tables, redundancy may be introduced in the resulting table, too. The extended definitions of projection and union are quite similar to the regular projection and union, except how to handle the redundancy. If two tuples are redundant in the resulting table, we combine these two tuples into one with new CL values which is disjunctively combined CL values from the original tuples.

A_3	A_4	CL
x_3	x_4	c_x'
w_3	w_4	w_y

Figure 9 : Table T_3

A_3	A_4	CL
x_3	x_4	$\delta(c_x,c_x',0,\wedge)$

Figure 10: T_2 ***intersect*** T_3

A_1	A_2	A_3	CL
u_1	u_2	a	c_u
v_1	v_2	a	c_v
w_1	w_2	b	c_w
x_1	x_2	b	c_x
y_1	y_2	b	c_y
z_1	z_2	z_3	c_z

Figure 11 : T_4

A_3	CL
a	$\delta(c_u, c_v, 0, \vee)$
b	$\delta(\,\delta(c_w, c_x, 0, \vee), c_y, 0, \vee)$
z_3	c_z

Figure 12 : $T_4\,[A_3]$

Before presenting the definition of an extended projection, let's define a function $SumCL$ which computes the CL value from a set of redundant tuples where the CL value is the disjunctive combination of all the CL values of the tuples.

Definition 4.10: Let Σ be a collection of sets of compatible tuples. Then a function $SumCL : \Sigma \rightarrow [0,1] \times [0,1]$ is defined as

$$SumCL(T \cup \{t\}) = \text{if } T = \{\,\} \qquad /\!*\ T \text{ is empty } *\!/$$
$$\text{then } t.CL$$
$$\text{else } \delta(t.CL, SumCL(T), 0, \vee).$$

The function $SumCL$ is used to compute the CL value for a tuple which replaces redundant tuples. Now, the definition of extended projection is given as follows.

Definition 4.11: For a non-empty table T and a set S of attributes where $S \neq \emptyset$ and $S \subset A(T)$, let π be a partition of a table T such as

$$\pi = \{\ \{\, t \mid (\exists t')_{t' \in T} (\forall a)_{a \in S} [t.a = t'.a]\ \} \mid t \in T\ \}.$$

Let π be denoted as $\{T^1, \ldots, T^n\}$ with $n = |\pi|$. Then, the projection of T onto S is defined as

$$T[S] = \ \bigcup_{i=1}^{n} \{\, t \mid A(t) = S \wedge t.CL = SumCL(T^i)$$
$$\wedge (\exists t')_{t' \in T^i} (\forall a)_{a \in S} [t.a = t'.a]\}.$$

The partition π of a table T is a collection of sets of redundant tuples when the table is projected on a set of attributes S. The resulting table from the extended projection consists of tuples with the new combined CL value from each set of redundant tuples. Let's look at an example, T_4 $[A_3]$ in Figure 12, where T_4 is shown in Figure 11.

Now, the definition of an extended union operation is following.

Definition 4.12: For any two tables T_1 and T_2 where $A(T_1) = A(T_2)$, the union of T_1 and T_2 is defined as

$$
\begin{aligned}
&T_1 \textbf{\textit{ union }} T_2 = \\
&\{t \mid A(t) = A(T_1) \wedge (\exists t_1)_{t1 \in T1}(\exists t_2)_{t2 \in T2}(\forall a)_{a \in A(T1)} [\, t_1.a = \\
&\qquad t_2.a \wedge t.a = t_2.a \wedge t.CL = \delta(t_1.CL, t_2.CL, 0, \vee)]\} \\
&\cup \{t \mid A(t) = A(T_1) \wedge t \in T_1 \\
&\qquad\qquad \wedge \neg(\exists t_2)_{t2 \in T2}[(\forall a)_{a \in A(T1)}[t_1.a = t_2.a]\} \\
&\cup \{t \mid A(t) = A(T_2) \wedge t \in T_2 \\
&\qquad\qquad \wedge \neg(\exists t_1)_{t1 \in T1}[(\forall a)_{a \in A(T2)}[t_1.a = t_2.a]\}
\end{aligned}
$$

When T_1 and T_2 have a duplicate tuple, T_1 ***union*** T_2 includes this tuple with disjunctively combined *CL* value from the corresponding *CL* values of tuples in T_1 and T_2. The set of these tuples constitutes the first set in the definition of T_1 ***union*** $T_{2.}$. The other two sets represent the non-duplicate tuples from T_1 and T_2. Let's look at an example, T_2 ***union*** T_5 in Figure 14.

4.4. Some Comments on the Extended Relational Algebra

The extended version of relational algebra is a generalization of the conventional relational algebra. When we have an atomic value for all the attribute values and [1 ; 1] for all the *CL* values in every table, the resulting database becomes a conventional one. We can easily prove that the conventional relational operators are just special cases of the relational operators defined above, except Join. Because Join is defined using Cartesian product and Select operation in the different way from the conventional one, it cannot be reduced to the conventional one directly.

The relational operators defined in this paper depend on the conjunctive combination and disjunctive combination of *Bel* and *Pls* values of two events. The Boolean combinations of *Bel* and *Pls* values are affected by the dependency relation between the two events. We made implicit independence assumption in defining relational operators for the simplicity of explanation. But, we can relax this condition in various ways. According to [Lee 92], the following inequality relations always hold.

$$
\begin{aligned}
Bel(A) * Bel(B) &\le Bel(A \wedge B) \le \min(Bel(A), Bel(B)) \\
Pls(A) * Pls(B) &\le Pls(A \wedge B) \le \min(Pls(A), Pls(B)) \\
1 - (1 - Bel(A)) * (1 - Bel(B)) & \\
&\ge Bel(A \vee B) \ge \max(Bel(A), Bel(B)) \\
1 - (1 - Pls(A)) * (1 - Pls(B)) & \\
&\ge Pls(A \vee B) \ge \max(Pls(A), Pls(B))
\end{aligned}
$$

The left-hand side values of the inequalities are used for the independent events, while the right-hand side values are used for the maximally dependent events. Otherwise, the belief of the conjunction of two dependent events can be determined by interpolating between the independent and the maximally dependent cases. When the dependency factor is not available, either the independence assumption can be made as in many probabilistic approaches, or the lower bound value for *Bel* value and the upper bound value for *Pls* value can be used. These are mostly domain-dependent decisions. But, there are some cases we can get that information. For example, ($T_{Applicants}$ ***where*** (STATUS = {GR})) ***union*** ($T_{Applicants}$ ***where*** (STATUS ≠ {GR})), and $T_{Managers}$ ***intersect*** $T_{Employees}$. For the first example, we can get the dependency relation from the query itself. For the second example, we can utilize the semantics of these two tables, since all the managers are also employees. The discussion of these problems is beyond the scope of this paper.

Because of the underlying representation (probability distribution of the power set of a domain), these relational operations have potentially very high time complexity. The time complexity of the relational operations basically depends on the number of focal elements in the data representation. But, when we consider that Dempster-Shafer theory is motivated by the practical reason that probability distribution is hardly available in the real world, when we consider that a basic probability assignment in Dempster-Shafer theory is one way to represent *incomplete* probability distribution (in the sense of the conference example), when we consider that if all focal elements are singleton the corresponding *Bel* function becomes *Bayesian function*, the number of focal elements in any data is not expected to be high. There are many applications where we can constrain the maximum number of focal elements to be two[6]. Since there is a trade-off between expressive power (number of focal elements) and efficiency, deciding the level of expressive power should be a domain-dependent

A_3	A_4	CL
g_3	g_4	c_g
x_3	f_4	c_f
y_3	y_4	c_h
d_3	d_4	c_d

Figure 13 : T_5

A_3	A_4	CL
x_3	x_4	c_x
y_3	y_4	$\delta(c_y, c_h, 0, \vee)$
g_3	g_4	c_g
x_3	f_4	c_f
d_3	d_4	c_d

Figure 14 : T_2 ***union*** T_5

[6] There are many cases where simple support functions (special case of belief function) are enough. Simple support function satisfies the following: for any non-empty set $A \subseteq D$, $m(A) = s$, $m(D) = 1 - s$, $m(elsewhere) = 0$.

decision. For the time complexity analysis and the efficient implementation of Dempster-Shafer theory, refer to [Shafe87], [Orpon90], [Guan 91].

5. Summary and Future Work

Humans are often expected to make decisions for their future based on available information that are usually incomplete and uncertain in nature. Fuzzy set approaches fail to handle stochastic data which are typical data representations in many fields. Due to a lot of restrictions, probability theory cannot be a practical tool in designing database systems. We chose Dempster-Shafer theory as a basic tool in our database model because of its flexible expressive power and sound theoretical foundation.

In this paper, we have proposed the new definitions of *Bel* and *Pls* functions which compute the belief of various comparisons between two independent basic probability distributions. These new definitions are used to define the select operation of the form $a \ \theta \ a'$. Based on the relational database model [Lee 92], we have defined five relational operations such as Select, Intersect, Union, Cartesian product, Projection and Join. The time complexity of the relational operators and the performance results of our ideas on an experimental prototype system will be reported in a forthcoming paper. There are several interesting problems for future research. One of them is to analyze how the dependency factors affect these relational operations and how to derive them from the query language and the domain knowledge in databases. Another is to find out how we formulate integrity constraints and functional dependencies in this model.

6. Acknowledgments

The presentation of this paper is made possible partially by the National Science Foundation Grant No. CCR-9202838. The author sincerely thanks Professor Hantao Zhang for discussions on the work reported here and helpful comments on earlier drafts of this paper. The author also appreciates Stuart Hansen and Jeff Line for proofreading an earlier draft of this paper.

7. References

[Abel 88] Abel, S. : "The sum-and-lattice-points method based on an evidential-reasoning system applied to the real-time vehicle guidance problem," *Uncertainty in Artificial Intelligence 2*, (1988) 365-370.

[Date 86] Date C.J.: *An Introduction to Database Systems, Vol. I* (4th ed). Addison Wesley, reading Mass., (1986).

[Duboi86] Dubois, D. and Prade, H.: *Possibility theory: An Approach to Computerized Processing of Uncertainty*, (1986) Plenum Press, New York.

[Duboi87] Dubois, D. and Prade, H.: "An Approach to Approximate Reasoning Based on the Dempster Rule of Combination," *International Journal of Expert Systems*, Vol. 1, No. 1 (1987) 67-85.

[Falke88] Falkenhainer, B. : "Towards a general-purpose belief maintenance system," *Uncertainty in Artificial Intelligence 2* (1988) 125-131

[Guan 91] Guan, J. and Bell, D. : *Evidence Theory and its Applications*, (1991) North-Holland.

[Halpe90] Halpern, J.Y. and Fagin, R.: "Two views of belief: Belief as generalized probability and belief as evidence," *Proc. AAAI-90* (1990) 112-119.

[Lee 92] Lee, S.K. : "Imprecise and Uncertain Information in Databases: An Evidential Approach," In *Proc. 8th Int. Conf. Data Engineering* (1992) 614-621.

[Lipsk79] Lipski, W.: "On Semantic Issues Connected with Incomplete Information Databases", *ACM Trans. Database Syst.* Vol. 4, No.3 (Sept. 1979) 262-296.

[Li 88] Li, Z. and Uhr, L. : "Evidential reasoning in a computer vision system," *Uncertainty in Artificial Intelligence 2* (1988) 403-412.

[Ola 92] Ola, A. : "Relational Databases with Exclusive Disjunction," In *Proc. 8th Int. Conf. Data Engineering* (1992) 328-335.

[Orpon90] Orponen, P.: "Dempster's rule of combination is #P-complete," *Artificial Intell.*, Vol. 44, (1990) 245-253.

[Prova90] Provan, G.M. : "The Application of Dempter-Shafer Theory to a Logic-Based Visual Recognition System," *Uncertainty in Artificial Intelligence 5* (1990) 389-406.

[Shafe76] Shafer, G.: *A Mathematical theory of evidence.* Princeton, NJ: Princeton Univ. Press, (1976).

[Shafe86] Shafer, G.: "Probability Judgment in Artificial Intelligence," *Uncertainty in Artificial Intelligence*, North-Holland (1986) 127-135.

[Shafe87] Shafer, G. and Logan, R.: "Implementing Dempster's rule for hierarchical evidence," *Artificial Intell.*, Vol. 33, (1987) 271-298.

[Willi88] Williams, M.H. and Nicholson, K.A., "An Approach to Handling Incomplete Information in Databases", *The Computer Journal*, Vol 31, No 2 (1988) 133-140.

[Zadeh78] Zadeh, L. A.: "Fuzzy sets as a basis for a theory of possibility," *Fuzzy Sets Syst.* vol. 1, no. 1 (1978) 3-28.