Implementing the Division Operation on a Database Containing Uncertain Data

FRANK S.C. TSENG
Department of Computer Science and
Information Engineering
National Chiao Tung University
Hsinchu, Taiwan 300, R.O.C.

ARBEE L.P. CHEN*
Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan 300, R.O.C.
Email: alpchen@cs.nthu.edu.tw
Fax: 886-35-723694

WEI-PANG YANG
Department of Computer Science and
Information Engineering
National Chiao Tung University
Hsinchu, Taiwan 300, R.O.C.

ABSTRACT

Uncertain data in databases were originally denoted as *null values*, which were later generalized to partial values. Based on the concept of partial values, we have further generalized the notion to *probabilistic partial values*. In this paper, an important operation — division, is fully studied to handle partial values and probabilistic partial values. Due to the uncertainty of partial values and probabilistic partial values, the corresponding extended division may produce *maybe tuples* and *maybe tuples with degrees of uncertainty*, respectively.

To process this extended division, we decompose a relation consisting of partial values or probabilistic partial values into a set of relations containing only definite values. Bipartite graph matching techniques are then applied to develop efficient algorithms for the extended division that handles partial values. The refinement on the maybe result is also discussed. Finally, we study the extended division that handles probabilistic partial values.

keywords: partial values, probabilistic partial values, graph matching, uncertain data.

*To whom all correspondence should be sent.

Contents

1 Introduction **1**

2 Division for Definite Data **2**

3 Generalizing Division to Handle Data with Partial Values **4**

3.1 Preliminaries for Partial Values 4

3.2 The Extended Division for Handling Partial Values 7

3.3 Preliminaries for Graph Theory 10

3.4 Algorithms for the Extended Division "$\hat{\div}$" 11

3.5 An Example 15

3.6 Further Refinement on Maybe Result 18

3.7 Refining Maybe Answers into True Answers 20

4 Generalizing Division to Handle Data with Probabilistic Partial Values **21**

4.1 Preliminaries for Probabilistic Partial Values 21

4.2 The Extended Division for Handling Probabilistic Partial Values 25

5 Discussion and Conclusion **26**

1 Introduction

Management of uncertain data has been recognized as an important research area in database systems [29]. *Null values* were originally adopted to represent the meaning of "values unknown at present" in database systems. Codd [12] pioneers the work on extended relational algebra to manipulate null values. From then on, incomplete information in relational databases have been extensively studied [5, 18, 21, 24, 25, 26]. The update semantics with null values in relational databases have been discussed in [1, 2]. Codd [13, 14] distinguishes null values into *applicable* and *inapplicable* values. Besides, the relationship between null values and functional dependencies has been studied in [22, 23, 39, 40]. [27] gives a concise review of handling null values by algebraic approaches.

The null value concept has been generalized to the concept of *partial values* by Grant [19]. Instead of being treated as an atomic value, an attribute value in a table is considered as a nonempty subset of the corresponding domain. A partial value in Grant [19] is represented as an interval such that exactly one of the values in the interval is the "true" value of the partial value. In DeMichiel [16], however, a partial value is considered to correspond to a finite set of *possible* values such that exactly one of the values in that set is the "true" value of the partial value. Motro [29] refers to this kind of partial values as *disjunctive data*. Therefore, an applicable null value is a partial value which corresponds to the whole domain. Lipski [24] provides a general discussion on manipulating uncertain information including partial values.

Following the notion of disjunctive data, we discuss the elimination of redundant partial values in [36], aggregate evaluation in [8] and the relationships among disjunctive data in [9][10][11]. Furthermore, in [37][38][33][34], we generalize partial values to *probabilistic partial values* and provide probabilistic approaches to query processing in heterogeneous database systems.

Among the previous research works and current commercial database systems, an important operation — the division, has not been taken into account for discussion and direct support. However, as we have pointed out in [35], this operation is needed for dealing with a query involving universal quantifiers. For example, based on the Suppliers-and-Parts database in [15], the query

"Get supplier numbers for suppliers who supply *all* parts."

corresponds to $\pi_{sno,pno}(\text{Shipments}) \div \pi_{pno}(\text{Parts})$ (refer to [15, Section 6.5.9]). Since it is not a *primitive* operator in the relational model, we need to implement the division by other primitive operators, which can be very cumbersome.

In the following sections, we first define the division operation for definite data and give an algorithm to directly implement this operation is given in Section 2. Then, we define the extended division for handling partial values and generalize the algorithm in Section 3. This generalized algorithm is developed by employing the bipartite graph matching technique [7]. In Section 4, we study the extended division for probabilistic partial values. Finally, Section 5 concludes our work.

2 Division for Definite Data

Let $r(X, Y)$ and $s(Y)$ be relations. The result of $r \div s$ can be defined to be the relation q as follows [27]:

$q(X) = \{t \mid$ for every tuple $t_s \in s$, there is a tuple $t_r \in r$ such that $t_r.X = t$ and $t_r.Y = t_s\}$.

This definition can also be specified as:

$$q(X) = \{t \mid (\{t\} \times s) \subseteq r\}.$$

In the following, an algorithm that directly implements this definition will be presented. A definition is needed for the algorithm and is stated as follows.

DEFINITION **2.1** For a relation $r(X, Y)$, X and Y possibly composite, the *grouping* of r by X, denoted $\overset{X}{\Delta}(r)$, is an operation that groups $r(X, Y)$ by X into $\gamma(X, Y)$ such that

$$\overset{X}{\Delta}(r) \equiv \gamma(X, Y) \equiv \{(x, \Upsilon(x)) \mid (x \in r.X) \wedge (\forall t \in r)((t.X = x) \Rightarrow t.Y \in \Upsilon(x))\}.$$

Notice that $\gamma(X, Y)$ is also called a *nested relation* [30].

For example, consider the following relation $r(X, Y)$

r

X	Y
a	x
a	z
d	w
d	x
d	z

Then, $\overset{X}{\Delta}(r)$ is

γ

X	Y
a	{x, z}
d	{w, x, z}

We give the algorithm as follows.

ALGORITHM **2.1** : An Algorithm That Derives the Result of $r(X,Y) \div s(Y)$.

Input: relations r(X, Y) and s(Y) without redundant tuples, X and Y possibly composite.

Output: the result of $r(X,Y) \div s(Y)$.

1. Ans = Ø; /* initialize Ans to be an empty set */
2. $r'(X,Y) = r(X,Y) \ltimes s(Y)$, where " $\ltimes$" denotes natural semijoin;
3. $\gamma(X,Y) = \overset{X}{\Delta}(r'(X,Y))$; i.e. group $r'(X,Y)$ by X into $\gamma(X,Y)$ — a nested relation.
4. for each tuple $t_i \in \gamma$ do {
5. if ($| t_i.Y |==| s |$) Ans = Ans $\cup$ $\{t_i.X\}$;
6. }
7. Return(Ans);

□

The natural semijoin in Step 2, $r'(X,Y) = r(X,Y) \ltimes s(Y)$, can be considered as a generalization of selection [4]; it restricts r by the values that appear in s. That is, $r' = \{t \mid t \in r \wedge t.Y \in s\}$. By this, we have the following lemma.

LEMMA **2.1** *For all $t_i \in \gamma$ in Algorithm 2.1, $| s | \geq \max_{\forall i}\{ | t_i.Y | \}$; that is, ($| t_i.Y | > | s |$) can never occur.*

Proof: Suppose there is a tuple $t_i \in \gamma$ such that ($| t_i.Y | > | s |$) then, by Step 3, there exists a tuple $t \in r'$ with $t.Y \notin s$, a contradiction to $r' = \{t \mid t \in r \wedge t.Y \in s\}$. Hence, ($| t_i.Y | > | s |$) never occurs and our lemma follows. □

We show the correctness of Algorithm 2.1 in the following.

THEOREM **2.1** *Algorithm 2.1 is correct.*

Proof: We denote the result produced by Algorithm 2.1 as A_1 and the result of $r(X,Y) \div s(Y)$ as A_2. By definition, $A_2 = \{t \mid (\{t\} \times s) \subseteq r\}$. We want to show that $A_1 = A_2$.

If $t \in A_1$ then there is a tuple $\tau \in \gamma$ such that

$$\tau.X \overset{Step5}{=} t \in \gamma.X \subseteq r'.X \subseteq r.X \text{ and } \tau.Y \overset{Step5}{=} \{y_1, y_2, \ldots, y_{|s|}\} \overset{Step2}{=} s.$$

Thus, we have $\{t\} \times s = \{\tau.X\} \times \tau.Y \overset{Step3}{\subseteq} r' \overset{Step2}{\subseteq} r$. Therefore, $t \in A_2$. Hence, $A_1 \subseteq A_2$.

Conversely, if $t \in A_2$ then $\{t\} \times s \subseteq r$. By Step 2 of Algorithm 2.1, we know $\{t\} \times s \subseteq r'$ and, by Step 3, there exists a tuple $\tau \in \gamma$ such that $\tau.X = t$ and $\tau.Y = s$. Therefore, $\mid \tau.Y \mid = \mid s \mid$ and t will be collected into Ans and returned by Algorithm 2.1 in Step 5 and 7, respectively; that is, $t \in A_1$. Hence $A_2 \subseteq A_1$. By the above discussion, we conclude that $A_1 = A_2$. □

Although the division operation for definite data can be implemented by some primitive operators of relational algebra. In Section 3, however, we will show the extended division operation over partial values cannot be implemented using DeMichiel's definitions of primitive operations [16]. Therefore, instead of using the primitive operators, Algorithm 2.1 will be used as a basis for generalizing division to deal with partial values.

3 Generalizing Division to Handle Data with Partial Values

3.1 Preliminaries for Partial Values

Partial values model data uncertainty in databases in the sense that, for an uncertain datum, its *true* value can be restricted in a specific set of possible values [16] or an interval of values [19]. In our work, a partial value is represented by a set of *possible* values, in which exactly one of the value is *true*. It is formally defined as follows.

DEFINITION **3.1** A *partial value*, denoted $\eta = [a_1, a_2, \ldots, a_n]$, associates with n possible values, $a_1, a_2, \ldots, a_n$, $n \geq 1$, of the same domain, in which exactly one of the values in η is the "true" value of η.

For a partial value $\eta = [a_1, a_2, \ldots, a_n]$, a function ν is defined on it in [16], where ν maps the partial value to its corresponding finite set of *possible* values; that is, $\nu(\eta) =$

$\{a_1, a_2, \ldots, a_n\}$. Recall that an *applicable null value* [13], $\aleph$, can be considered as a partial value with $\nu(\aleph) = D$, where D is the whole domain. Besides, a relation containing partial values is called a *partial relation* [16]. Otherwise, it will be called a *definite relation*. Except for explicitly specified, a relation is considered to be *partial* hereafter. In the following, we will use η and $\nu(\eta)$ interchangeably when it does not cause confusion. For example, $v \in \eta$ if $v \in \nu(\eta)$.

The *cardinality* of a partial value η is defined as $\mid \nu(\eta) \mid$ in [16]. When the cardinality of a partial value equals to 1, i.e., there exists only one *possible* value, say d, in the partial value, then the partial value $[d]$ actually corresponds to the definite value d. On the other hand, a definite value d can be represented as a partial value $[d]$. Besides, a partial value with cardinality greater than 1 is referred to as a *proper partial value* in [16]. For any two proper partial values, say η_1 and η_2, $\eta_1 \neq \eta_2$ even if $\nu(\eta_1) = \nu(\eta_2)$. This is because the *true* value of η_1 may not be the same as the *true* value of η_2.

DEFINITION **3.2** If the proper partial values, $\eta_1, \eta_2, \ldots, \eta_k, k \geq 2$, are elements of a set of partial values, Φ, and $\nu(\eta_1) = \nu(\eta_2) = \cdots = \nu(\eta_k)$, then we say $\eta_1, \eta_2, \ldots, \eta_{i-1}, \eta_{i+1}, \ldots, \eta_k$ are *quasi-duplicates* of $\eta_i, 1 \leq i \leq k$.

By Definition 3.2, if $\Phi = \{\overbrace{[a,b]}^{\eta_1}, \overbrace{[a,b]}^{\eta_2}\}$ then η_1 is a quasi-duplicate of η_2, and vice versa.

When a set of attributes $\{A_1, A_2, \cdots, A_m\}$ is considered as a *composite attribute* $A_1 A_2 \cdots A_m$, its values can be computed from the values in this set of attributes as follows.

A_1	A_2	$\cdots$	A_m
η_{11}	η_{21}	$\cdots$	η_{m1}
η_{12}	η_{22}	$\cdots$	η_{m2}
$\vdots$	$\vdots$	$\ddots$	$\vdots$
η_{1n}	η_{2n}	$\cdots$	η_{mn}

$\Longleftrightarrow$

$A_1 A_2 \cdots A_m$
$\eta_{11} \hat{\times} \eta_{21} \hat{\times} \cdots \hat{\times} \eta_{m1}$
$\eta_{12} \hat{\times} \eta_{22} \hat{\times} \cdots \hat{\times} \eta_{m2}$
$\vdots$
$\eta_{1n} \hat{\times} \eta_{2n} \hat{\times} \cdots \hat{\times} \eta_{mn}$

The $\hat{\times}$ denotes the cartesian product of partial values, which is defined as follows.

DEFINITION **3.3** The *cartesian product* $\eta_a \hat{\times} \eta_b$ of the partial values $\eta_a = [a_1, a_2, \ldots, a_m]$ and $\eta_b = [b_1, b_2, \ldots, b_n]$ is the partial value $\eta_{a \hat{\times} b}$ with $\nu(\eta_{a \hat{\times} b})$ being a set of the ordered pairs (a_i, b_j) for every $a_i \in \eta_a$ and $b_j \in \eta_b$.

We regard $(\eta_{1j}, \eta_{2j}, \ldots, \eta_{mj})$ and $\eta_{1j} \hat{\times} \eta_{2j} \hat{\times} \ldots \hat{\times} \eta_{mj}$ as semantically-equivalent because if the "true" value of $(\eta_{1j}, \eta_{2j}, \cdots, \eta_{mj})$ is the m-tuple $(a_1, a_2, \ldots, a_m)$, where $a_i \in \eta_{ij}$ and is

the "true" value of η_{ij}, then the "true" value of $\eta_{1j}\hat{\times}\eta_{2j}\hat{\times}\cdots\hat{\times}\eta_{mj}$ is also $(a_1, a_2, \ldots, a_m)$, and vice versa.

EXAMPLE **3.1** Consider the following data values in the set of attributes $\{A_1, A_2\}$

A_1	A_2
1	$[a, b]$
$[2, 3]$	c

We can regard $\{A_1, A_2\}$ as a composite attribute A_1A_2 and compute its data as

A_1A_2
$[(1, a), (1, b)]$
$[(2, c), (3, c)]$

□

A partial relation $R(A_1, A_2, \ldots, A_m)$ therefore can be regarded as a relation $R(A_1A_2\cdots A_m)$ with the composite attribute $A_1A_2\cdots A_m$. That is, a partial relation can be considered as a set of partial values. By this, we have the following definitions.

DEFINITION **3.4** An *interpretation*, $\alpha = (a_1, a_2, \ldots, a_m)$, of a set of partial values, $\Phi = \{\eta_1, \eta_2, \ldots, \eta_m\}$, is an assignment of values from Φ such that $a_i \in \eta_i, 1 \leq i \leq m$.

By Definition 3.4, for a set of partial values $\Phi = \{\eta_1, \eta_2, \ldots, \eta_m\}$, $\nu(\eta_1)\times\nu(\eta_2)\times\cdots\times\nu(\eta_m)$ is the set of all interpretations of Φ.

DEFINITION **3.5** For an interpretation $\alpha = (a_1, a_2, \ldots, a_m)$ of a set of partial values $\Phi = \{\eta_1, \eta_2, \ldots, \eta_m\}$, the *value set* of α is denoted $S_\alpha = \bigcup_{1\leq i\leq m}\{a_i\}$.

DEFINITION **3.6** For all interpretations, α_j, $1 \leq j \leq p$, $p = \mid \eta_1 \mid \times \mid \eta_2 \mid \times\cdots\times \mid \eta_m \mid$, of a set of partial values $\Phi = \{\eta_1, \eta_2, \ldots, \eta_m\}$, the *family of value sets* of Φ is denoted $\mathcal{F}(\Phi) = \bigcup_{1\leq j\leq p}\{S_{\alpha_j}\}$. If $\Phi = \emptyset$ then define $\mathcal{F}(\Phi) = \emptyset$.

EXAMPLE **3.2** Consider the following relation $r(X, Y)$.

r

X	Y
a	y
a	$[x, y]$

We can regard $\{X, Y\}$ as a composite attribute XY and compute r to be

r

XY
$[(a,y)]$
$[(a,x),(a,y)]$

Then there are two interpretations, α_1 and α_2, of r:

$$\alpha_1 = ((a,y),(a,x)) \quad \text{and} \quad \alpha_2 = ((a,y),(a,y)).$$

Their corresponding value sets are

$$S_{\alpha_1} = \{(a,y)\} \cup \{(a,x)\} = \{(a,y),(a,x)\} \quad \text{and} \quad S_{\alpha_2} = \{(a,y)\} \cup \{(a,y)\} = \{(a,y)\}.$$

Therefore, the family of value sets of r is

$$\begin{aligned} \mathcal{F}(r) &= \{S_{\alpha_1}\} \cup \{S_{\alpha_2}\} \\ &= \{\{(a,y),(a,x)\},\{(a,y)\}\}, \end{aligned}$$

which corresponds to the following two definite relations:

S_{α_1}

X	Y
a	y
a	x

S_{α_2}

X	Y
a	y

3.2 The Extended Division for Handling Partial Values

In [16], DeMichiel presents a set of algebraic operations over partial values. However, the extended division for partial values is not studied. For relations containing partial values, due to the uncertainty of partial values, the result of an extended division may contain *maybe tuples* [5, 12, 16].

For a division operation with operands $A(X,Y)$ and $B(Y)$ of *definite* data, the following equation always holds.

$$A(X,Y) \div B(Y) = \pi_X(A) - \pi_X((\pi_X(A) \times B) - A)$$

That is, the division operation for definite data can be implemented by some primitive operators of relational algebra. However, in the following, we will show the extended division operation over partial values cannot be implemented using the above definition and DeMichiel's definitions of primitive operations. That is,

$$A(X,Y) \hat{\div} B(Y) \neq \pi'_X(A) -' \pi'_X((\pi'_X(A) \times' B) -' A),$$

where $\hat{\div}$ denotes the extended division for partial values and π', $-'$, and $\times'$ are as defined in [16].

LEMMA **3.1** *If $A(X,Y)$ and $B(Y)$ contain true tuples only, then*

$$\pi'_X(A) -' \pi'_X((\pi'_X(A) \times' B) -' A)$$

can never produce maybe results, where π', $-'$, and $\times'$ are as defined in [16].

Proof: By DeMichiel's Definition, the *maybe* result of $\pi'_X(A)$ is

$$\{t \mid (\exists u)(u \in A \wedge (status(u) = maybe) \wedge (t = u.X) \\ \wedge (\nexists v)(v \in A \wedge (v.X = u.X) \wedge (status(v) = true)))\}.$$

Since, by assumption, A has no maybe tuples, the maybe result of $\pi'_X(A)$ is therefore empty. Thus,

$$\begin{aligned} & \pi'_X(A) -' \pi'_X((\pi'_X(A) \times' B) -' A) \\ = \; & \emptyset -' \pi'_X((\emptyset \times' B) -' A) \\ = \; & \emptyset \end{aligned}$$

□

By Lemma 3.1, we know that if we perform the extended division $A(X,Y)\hat{\div}B(Y)$ by $\pi'_X(A) -' \pi'_X((\pi'_X(A) \times' B) -' A)$, then the maybe result tuples which should have been generated by the true tuples of the original relations $A(X,Y)$ and $B(Y)$ will be lost. We define $A(X,Y)\hat{\div}B(Y)$ as a primitive operation in the following. In our work, we will assume a divisor contains no partial values, otherwise the extended division will never produce true results.

The extended division for partial values is defined as follows. Let the source relations be $r(X,Y)$ and $s(Y)$, where X and Y are possibly composite. We denote the true result of $r \;\hat{\div}\; s$ as $\mathcal{TR}$.

$$\mathcal{TR} \equiv \bigcap_{S_{\alpha_i} \in \mathcal{F}(r)} (S_{\alpha_i} \div s) \equiv \{t \mid (t \in \bigcup_{u_i \in r} \nu(u_i.X)) \wedge (\{t\} \times s) \subseteq r\}.$$

The maybe result, denoted $\mathcal{MR}$, is defined as follows.

$$\mathcal{MR} \equiv \bigcup_{S_{\alpha_i} \in \mathcal{F}(r)} (S_{\alpha_i} \div s) - \mathcal{TR},$$

where $\div$ denotes the conventional division for definite relations.

The following example illustrates this definition.

EXAMPLE **3.3** Consider the relations r and s given below and suppose we want to compute $r(X,Y) \hat{\div} s(Y)$.

r

X	Y
a	x
a	z
$[b,c]$	x
$[b,c]$	z

s

Y
x
z

By the definition, we can easily derive the true result of $r(X,Y) \hat{\div} s(Y)$ to be $\{a\}$. To compute the maybe result, we first transform $r(X,Y)$ into the following equivalent relation $r(XY)$.

r

XY
$[(a,x)]$
$[(a,z)]$
$[(b,x),(c,x)]$
$[(b,z),(c,z)]$

There are four interpretations for r:

$$\begin{aligned} \alpha_1 &= ((a,x),(a,z),(b,x),(b,z)), \\ \alpha_2 &= ((a,x),(a,z),(b,x),(c,z)), \\ \alpha_3 &= ((a,x),(a,z),(c,x),(b,z)), \text{ and} \\ \alpha_4 &= ((a,x),(a,z),(c,x),(c,z)); \end{aligned}$$

and the corresponding value sets are shown below.

S_{α_1}

X	Y
a	x
a	z
b	x
b	z

S_{α_2}

X	Y
a	x
a	z
b	x
c	z

S_{α_3}

X	Y
a	x
a	z
c	x
b	z

S_{α_4}

X	Y
a	x
a	z
c	x
c	z

By these value sets, we have

$$S_{\alpha_1} \div s = \{a,b\}, \quad S_{\alpha_2} \div s = \{a\},$$
$$S_{\alpha_3} \div s = \{a\}, \quad \text{and} \quad S_{\alpha_4} \div s = \{a,c\}.$$

That is, $\bigcup_{S_{\alpha_i} \in \mathcal{F}(r)} (S_{\alpha_i} \div s) = \{a,b,c\}$. Therefore, the result q of $r \hat{\div} s$ is

q

X	status
a	true
b	maybe
c	maybe

Note that an extra *status* column is added for the result of an extended division to distinguish between true and maybe tuples. It does not correspond to a real attribute. □

However, from the above definition, it is not a trivial task to determine the result of an extended division. Since the interpretations of a partial relation r can be very large (refer to Definition 3.4), we need a more efficient algorithm for the extended division. In the following, we apply bipartite graph matching techniques to develop an algorithm for solving this problem.

3.3 Preliminaries for Graph Theory

In the following, some terminologies about *graph theory* are defined [7, 6]. A *graph* G is denoted $G = (V, E)$, where V is the set of *vertices* and E is the set of *edges* in the graph. An edge (x, y) is said to join the vertices x and y. If $(x, y) \in E$ then x and y are *adjacent* or *neighboring* vertices of G. For any set $S \subseteq V$, we define the *neighbor set* of S in G, denoted $N(S)$, to be the set of all vertices adjacent to the vertices in S. A *bipartite graph* $G = (V, E)$ is one whose vertex set V can be partitioned into two subsets X and Y, such that each edge in G joins a vertex in X and a vertex in Y. If two vertices are not joined by an edge, they are said to be *independent*. Similarly, two edges that do not share a common vertex are said to be *independent*. Given a subset S of V, the subgraph *induced* by S, denoted $G[S]$, is the graph with vertex set S and edge set containing those edges of G joining two vertices of S. A *clique* of G is a subset S of V such that $G[S]$ is a *complete graph*, that is, each pair of distinct vertices in $G[S]$ is joined by an edge. A set of pairwise independent edges is called a *matching*. A matching of maximum cardinality is called a *maximum matching*. Besides, two more terms are defined as follows.

DEFINITION **3.7** Let $S = \{S_1, S_2, \ldots, S_n\}$ be a family of sets and $s = \{s_1, s_2, \ldots, s_m\}$. The *membership graph of S over s* is a bipartite graph $G = (V, E) = (X \cup Y, E)$, where

$$X = \mathrm{s} \ = \{s_1, s_2, \ldots, s_m\},$$

$Y = \mathrm{S} = \{S_1, S_2, \ldots, S_n\}$, and

$E = \{(s_i, S_j) \mid s_i \in S_j, 1 \leq i \leq m, 1 \leq j \leq n\}$.

DEFINITION **3.8** For a bipartite graph $G = (X \cup Y, E)$, $\mid X \mid \leq \mid Y \mid$, we say that there is a *complete matching M from X to Y* if there is a matching of cardinality $\mid X \mid$; that is, each vertex in X is adjacent to a distinct vertex in Y.

3.4 Algorithms for the Extended Division "$\hat{\div}$"

In this section, we present two C-like algorithms to derive the (true and maybe) answer of an extended division operation. In the first place, Algorithm 2.1 will be slightly modified for obtaining the true result of an extended division then this algorithm will be generalized to derive the maybe result.

ALGORITHM **3.1** : An Algorithm That Derives the *True Result* of an Extended Division.

Input: relations $r(X, Y)$ and $s(Y)$, where X and Y are possibly composite and s contains no partial values.

Output: the true result of $r(X, Y) \hat{\div} s(Y)$.

1. TrueAns = $\emptyset$; /* initialize TrueAns to be an empty set */
2. $r' = r$ with all tuples that contain *proper* partial values eliminated;
3. Perform $r''(X, Y) = r'(X, Y) \ltimes s(Y)$, where " $\ltimes$" denotes semijoin;
4. $\gamma(X, Y) = \overset{X}{\Delta}(r''(X, Y))$;
5. for each tuple $t_i \in \gamma$ do {
6. if ($\mid t_i.Y \mid == \mid s \mid$) TrueAns = TrueAns $\cup$ $\{t_i.X\}$;
7. }
8. Return(TrueAns);

Except for Step 2, this algorithm is the same as Algorithm 2.1. Step 2 guarantees r' to be a definite relation. The following theorem verifies the correctness of Algorithm 3.1.

THEOREM **3.1** *Algorithm 3.1 correctly produces the true result of* $r(X, Y) \hat{\div} s(Y)$.

Proof: The proof is the same as that of Theorem 2.1, except that r and r' in the proof of Theorem 2.1 are replaced by r' and r'', respectively. □

Before presenting the algorithm that derives the maybe result for an extended division, we need some preliminary definitions.

DEFINITION **3.9** The *partial semi-join* of relation $r(X,Y)$ by relation $s(Y)$, written $r(X,Y) \propto s(Y)$, is an operation that produces the relation

$$r''(X,Y) = \{t \mid (\forall u \in r)((u.Y \cap s \neq \emptyset) \Rightarrow (t.X = u.X \wedge t.Y = (u.Y \cap s)))\}.$$

The following example illustrates this definition.

EXAMPLE **3.4** For the relations r and s given below,

r

X	Y
a	z
a	[w,u]
[b,c]	x
e	[x,y]
e	[y,z]

s

Y
x
z

the result q $= r(X,Y) \propto s(Y)$ is

q

X	Y
a	z
[b,c]	x
e	[x]
e	[z]

DEFINITION **3.10** For a relation $r(X,Y)$, X and Y possibly composite, the *partial grouping* of r by X, denoted $\overset{X}{\nabla}(r)$, is an operation that groups $r(X,Y)$ by X into $\gamma(X,Y)$; that is $\overset{X}{\nabla}(r) = \gamma$, where

$$\gamma(X,Y) = \{(x, \Upsilon(x)) \mid (x \in \textstyle\bigcup_{t_i \in r} \nu(t.X)) \wedge (\forall t \in r)((x \in \nu(t.X)) \Rightarrow t.Y \in \Upsilon(x))\}.$$ □

It is easy to verify that when $r(X,Y)$ is a definite relation, then $\overset{X}{\nabla}(r) = \overset{X}{\Delta}(r)$ (refer to Definition 2.1).

EXAMPLE **3.5** Let $r(X,Y)$ be the relation listed below.

r

X	Y
a	x
b	y
c	z
a	y
a	z
[a, c]	[w, x]
[a, b]	[x, z]

Then the partial grouping $\overset{X}{\nabla}(r)$ is

γ

X	Y
a	{x, y, z, [w, x], [x, z]}
b	{y, [x, z]}
c	{z, [w, x]}

We now discuss how a maybe result tuple of $r(X,Y) \mathbin{\hat{\div}} s(Y)$ can be derived. The following theorem specifies the necessary and sufficient condition for a true or maybe answer of an extended division.

THEOREM **3.2** *For a tuple $(x, \Upsilon(x))$ in $\gamma = \overset{X}{\nabla}(r)$, where $\Upsilon(x) = \{\eta_1, \eta_2, \ldots, \eta_n\}$, there exists a complete matching M from $s = \{s_1, s_2, \ldots, s_m\}$ to $S = \Upsilon(x) = \{\eta_1, \eta_2, \ldots, \eta_n\}$ in the membership graph $G = (s \cup S, E)$ if and only if x is an answer (true or maybe) of $r(X,Y) \mathbin{\hat{\div}} s(Y)$.*

Proof: ($\Leftarrow$) (1) If x is a true answer of $r(X,Y) \mathbin{\hat{\div}} s(Y)$ then $\{x\} \times s \subseteq r$. By regarding the elements of s as partial values with cardinality one, we have $s = \{[s_1], [s_2], \ldots, [s_m]\}$. Then, by Definition 3.10, there exists a tuple $(x, \Upsilon(x)) \in \gamma = \overset{X}{\nabla}(r)$, such that $s \subseteq \Upsilon(x) = S$. Hence, there is a complete matching $M = \{(s_1, [s_1]), (s_2, [s_2]), \ldots, (s_m, [s_m])\}$ from s to S in the membership graph $G = (s \cup S, E)$.

(2) If x is a maybe answer of $r(X,Y) \mathbin{\hat{\div}} s(Y)$ then $x \in \bigcup_{S_i \in \mathcal{F}(r)} (S_i \div s)$. That is, there exists an interpretation α with value set S_α, such that $\{x\} \times s \subseteq S_\alpha \in \mathcal{F}(r)$. Therefore, there is a tuple $(x, \Upsilon(x)) \in \gamma$ with $\Upsilon(x) = \{\eta_1, \eta_2, \ldots, \eta_m, \ldots\}$, such that $s_i \in \eta_i$, for $i = 1, 2, \ldots, m$. Hence, there is a complete matching $M = \{(s_1, \eta_1), (s_2, \eta_2), \ldots, (s_m, \eta_m)\}$ from s to S in the membership graph $G = (s \cup S, E)$.

($\Rightarrow$) If there exists a complete matching $M = \{(s_1, \eta_1), (s_2, \eta_2), \ldots, (s_m, \eta_m)\}$ in $G = (s \cup S, E)$, then $s_i \in \eta_i$, for $i = 1, 2, \ldots, m$. By Definition 3.10, r contains m tuples, $t_1, t_2, \ldots, t_m$, such that $x \in t_i.X$ and $\eta_i = t_i.Y$, $i = 1, 2, \ldots, m$. Therefore, there exists an interpretation $\alpha = ((x, s_1), (x, s_2), \ldots, (x, s_m), \ldots)$ of r with its value set $S_\alpha = \{(x, s_1), (x, s_2), \ldots, (x, s_m), \ldots\}$, which implies $\{x\} \times s \subseteq S_\alpha \in \mathcal{F}(r)$. Hence, $x \in (S_\alpha \div s) \subseteq \bigcup_{S_i \in \mathcal{F}(r)} (S_i \div s)$. When $\{x\} \times s \subseteq S_\alpha \subseteq r$ then x is a true result. Otherwise, x is a maybe result. □

In the following, we present Algorithm 3.2, which is generalized from Algorithm 3.1, to derive the maybe result of an extended division.

ALGORITHM **3.2** : An Algorithm That Derives the *Maybe Result* of an Extended Division Operation.

Input: relations $r(X, Y)$, $s(Y)$ and the TrueAns obtained from Algorithm 3.1, where X and Y are possibly composite and s contains neither partial values nor maybe tuples.

Output: the maybe result of $r(X, Y) \stackrel{\widehat{\div}}{} s(Y)$.

1. MaybeAns = $\emptyset$;
2. Perform $r''(X, Y) = r(X, Y) \propto s(Y)$, where "$\propto$" denotes partial semi-join;
3. $\gamma = \stackrel{X}{\nabla}(r'')$; i.e., apply the partial grouping operation to r'' by X;
4. for each tuple $t \in \gamma$ do {
5. if ($t.X \in$ TrueAns) continue; /* true result overrides maybe result */
6. if ($\mid t.Y \mid < \mid s \mid$) continue; /* prune impossible answer */
7. if ($\bigcup_{\eta_j \in t.Y} (\eta_j) \neq s$) continue; /* prune impossible answer */
8. if (the membership graph of $S = \{\eta_1, \eta_2, \ldots, \eta_n\}$ over s, $\forall \eta_j \in t.Y$, has a complete matching M from s to S){
9. MaybeAns = MaybeAns $\cup$ $\{t.X\}$;
10. }
11. }
12. Return(MaybeAns); □

Note that Step 5, Step 6 and Step 7 are used to prune impossible answers. In Step 5, a true result reasonably overrides a maybe result. In Step 6, it is obvious that if

$|\ t.Y\ | < |\ s\ |$ then $t.X$ cannot be an answer, since there is no complete matching. For Step 7, we claim that if $\bigcup_{\eta_j \in t.Y}(\eta_j) \neq s$ then $\bigcup_{\eta_j \in t.Y}(\eta_j) \subset s$ and $t.X$ can never be an answer. That is, $\bigcup_{\eta_j \in t.Y}(\eta_j) \supset s$ is impossible. In proof, if $\bigcup_{\eta_j \in t.Y}(\eta_j) \supset s$ then there exists an element in $\bigcup_{\eta_j \in t.Y}(\eta_j)$ but not in s, which implies there is a tuple $t \in r''$ such that $t.Y \not\subseteq s$ — a contradiction to the definition of partial semi-join as used in Step 2. Except for these steps (i.e., Steps 2, 5, 6, and 7), this algorithm actually implements Theorem 3.1. Therefore, its correctness is also verified.

For Step 8, an $O(n^{2.5})$ algorithm in [20], can be used to find a complete matching for a bipartite graph $G = (V, E) = (X \cup Y, E)$, where n is the number of nodes in G.

3.5 An Example

In the following, an example is presented to illustrate the whole processes of Algorithms 3.1 and 3.2.

EXAMPLE **3.6** Consider the following relations $r(X, Y)$ and $s(Y)$, where $r(X, Y)$ is wrapped for saving space. We apply Algorithms 3.1 and 3.2 to derive the true result and maybe result of $r(X, Y) \mathbin{\hat{\div}} s(Y)$, respectively.

r

X	Y	X	Y
a	w	[d, e, f]	[y, z]
a	x	[d, f]	[x, z]
a	y	e	[w, x]
a	z	e	x
[a, b]	w	e	w
b	x	f	[u, v, w, x]
b	y	f	[v, w]
b	z	g	w
c	[v, x, y]	g	x
c	[u, x, z]	[g, h]	y
c	[u, v, w, y]	[g, h]	z
d	[x, y]	h	w
d	[x, y]	h	[x, u]

s

Y
w
x
y
z

Recall that two proper partial values are not supposed to be equal even if they correspond to the same set of possible values. Therefore, the quasi-duplicate

d	[x, y]

cannot be eliminated, or information may be lost. (However, under some circumstances, some quasi-duplicates may be eliminated without losing information. We have explored this problem in [36].)

First, we apply Algorithm 3.1 to derive the true result. After Step 2, we get r' as:

r'

X	Y	X	Y	X	Y
a	w	b	x	e	w
a	x	b	y	g	w
a	y	b	z	g	x
a	z	e	x	h	w

After Step 3, r'' remains the same as r'. So, by Step 4, we group r'' by X into γ as:

γ

X	Y
a	{w, x, y, z}
b	{x, y, z}
e	{x, w}
g	{x, w}
h	{w}

For those tuples t's in γ, only the first tuple satisfies the condition of Step 6. Therefore, the true result is {a}. Now we proceed to Algorithm 3.2. For convenience, we treat all the attribute values in $r.Y$ as partial values (i.e., convert all definite values into partial values with cardinality one) and represent this new relation, r', as follows.

r'

X	Y	X	Y
a	[w]	[d, e, f]	[y, z]
a	[x]	[d, f]	[x, z]
a	[y]	e	[w, x]
a	[z]	e	[x]
[a, b]	[w]	e	[w]
b	[x]	f	[u, v, w, x]
b	[y]	f	[v, w]
b	[z]	g	[w]
c	[v, x, y]	g	[x]
c	[u, x, z]	[g, h]	[y]
c	[u, v, w, y]	[g, h]	[z]
d	[x, y]	h	[w]
d	[x, y]	h	[x, u]

By performing partial semi-join in Step 2, we obtain r'' as

r''

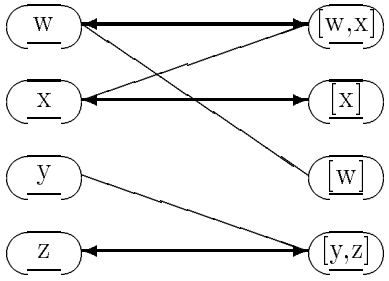
X	Y	X	Y
a	[w]	[d, e, f]	[y, z]
a	[x]	[d, f]	[x, z]
a	[y]	e	[w, x]
a	[z]	e	[x]
[a, b]	[w]	e	[w]
b	[x]	f	[w, x]
b	[y]	f	[w]
b	[z]	g	[w]
c	[x, y]	g	[x]
c	[x, z]	[g, h]	[y]
c	[w, y]	[g, h]	[z]
d	[x, y]	h	[w]
d	[x, y]	h	[x]

After applying the partial grouping operation, we obtain γ as follows.

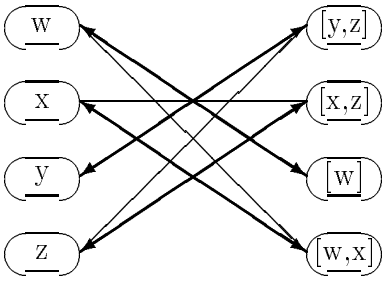
γ

	X	Y
1	a	{[w], [x], [y], [z]}
2	b	{[w], [x], [y], [z]}
3	c	{[x, y], [x, z], [w, y]}
4	d	{[x, y], [x, y], [y, z], [x, z]}
5	e	{[y, z], [w, x], [x], [w]}
6	f	{[y, z], [x, z], [w, x], [w]}
7	g	{[w], [x], [y], [z]}
8	h	{[w], [x], [y], [z]}

For those tuples in γ, the first tuple satisfies the condition in Step 5, the second, seventh, and eighth tuples satisfy the condition in Step 8. For the third and the fourth tuples, they satisfy the conditions in Steps 6 and 7, respectively. Therefore, they are not included into MaybeAns. Now, consider the fifth tuple, we find one of the maximum matchings as follows.



It does not satisfy the condition in Step 8. Finally, for the sixth tuple, we can find a complete matching as follows.



After performing the for loop between Steps 4 and 11, we obtain MaybeAns = {b, f, g, h}. □

3.6 Further Refinement on Maybe Result

In the maybe result obtained by Algorithm 3.2, there may exist some inter-relationships among these maybe result tuples. We say that two maybe result tuples α and β of an extended division $r(X,Y) \hat{\div} s(Y)$ are *strongly dependent* if there exists one tuple t in the original relation r which contributes to both α and β to be the maybe result tuples and whose deletion makes both α and β no longer be in the maybe result. Since there is only one "true" value in t, α and β cannot *coexist*. In Example 3.6, g and h are strongly dependent. This is because, for example, the following tuple in the original relation r

[g, h]	y

contributes to both g and h to be maybe result tuples and whose deletion makes *both* g and h unable to be in the maybe result.

If C is a subset of the maybe result of an extended division such that each pair of the elements in C are strongly dependent, then we say C is a *strongly dependent clique*. In a strongly dependent clique C, any two elements in C cannot coexist. Therefore, if $C = \{c_1, c_2, \ldots, c_n\}$ is a strongly dependent clique, then we can represent $\{c_1, c_2, \ldots, c_n\}$ as a partial value $[c_1, c_2, \ldots, c_n]$ and regard it as a maybe result tuple. That is, a maybe result tuple

$[c_1, c_2, \ldots, c_n]$	maybe

means that if this answer tuple exists then it must be exactly one of c_i in $[c_1, c_2, \ldots, c_n]$. In comparison, the original maybe result

c_1	maybe
c_2	maybe
$\vdots$	$\vdots$
c_n	maybe

means that the answer is $(c_1 \vee c_2 \vee \cdots \vee c_n) \oplus \emptyset$, where "$\vee$" is the inclusive-or operator and "$\oplus$" is the exclusive-or operator, which does not capture the dependency. For example, in the following operation

X	Y
[a, b]	[x, z]
[b, c]	[y, z]
[a, c]	[x, y]

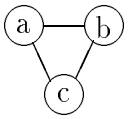
$\hat{\div}$

Y
x
z

$=$

X	status
a	maybe
b	maybe
c	maybe

These answer tuples are pairwise strongly dependent. Therefore, the maximal strongly dependent clique is



and the answer can be further refined into

[a, b, c]	maybe

To find a maximal strongly dependent clique in the maybe result $\mathcal{MR}$ of an extended division, we can construct a graph $G = (V, E)$ as follows:

1. $V = \{m \mid m \in \mathcal{MR}\}$
2. $E = \{(m_i, m_j) \mid m_i$ and m_j are strongly dependent$\}$.

Then find a maximal clique in G. Both the problems can be shown to be equivalent. That is, they can be transformed into each other. Unfortunately, finding a maximal clique on graphs is a well-known NP-hard problem [17].

3.7 Refining Maybe Answers into True Answers

We observe that, in some cases, some maybe result tuples can be further refined into a true result tuple. We present this idea by an example followed by its formal description.

EXAMPLE **3.7** Consider the following relations $r(X, Y)$ and $s(Y)$.

r

X	Y
a	x
[a, b, c]	y
b	x
c	x

s

Y
x
y

The answer of $q = r(X, Y) \hat{\div} s(Y)$ is

q

X	status
a	maybe
b	maybe
c	maybe

which can be refined into

q

X	status
[a, b, c]	true

This is because the second tuple of r is exactly one of (a, y), (b, y), and (c, y). □

This refinement on the maybe result of $r(X, Y) \hat{\div} s(Y)$ can be obtained when there is a tuple $t \in r$ with

1. $\nu(t.X) = \{m_1, m_2, \ldots, m_k\}$, $k \geq 2$, and
2. $t.Y$ contains a definite value only,

such that,

1. the deletion of t makes k maybe answer tuples, $m_1, m_2, \ldots, m_k$, no longer be in the maybe result, and
2. $(\{m_1, m_2, \ldots, m_k\} \times (s - t.Y)) \subset r$.

If this condition is satisfied, then these maybe answers, $m_1, m_2, \ldots, m_k$, can be grouped into a single partial value $[m_1, m_2, \ldots, m_k]$ with status *true*. This is because the true value of $t.X$ can be exactly one of m_i's, which implies there must be exactly one m_i satisfies $\{m_i\} \times s \subseteq r$.

The semantic meaning for an answer tuple with true/maybe status can be clarified as follows.

1. a tuple | [a, b] | maybe | means that either (i) it is either a or b or (ii) it is nothing.
2. a tuple | a | maybe | means that it is either a or nothing.
3. a tuple | [a, b] | true | means that it is either a or b.
4. a tuple | a | true | means that it is a, definitely.

4 Generalizing Division to Handle Data with Probabilistic Partial Values

Although we can refine maybe results into more informative answers, users have no way to distinguish which maybe tuple is the most possible answer. In [37][38], the concept of partial values is generalized to *probabilistic partial values*, and a probabilistic approach to query processing in heterogeneous database systems is provided. We employ the concept of probabilistic partial values to provide maybe answer tuples with degrees of uncertainty. This will make the comparison among maybe answer tuples possible, and provide more informative answers for users. In this section, we study the extended division for dealing with probabilistic partial values.

4.1 Preliminaries for Probabilistic Partial Values

We generalize partial values to probabilistic partial values by regarding an attribute as a discrete random variable [28, 3]. Sadri [31, 32] has developed a probabilistic approach to modeling uncertainty in databases. In Sadri's work, the degree of uncertainty is attached on tuple level. However, in our work, the degree of uncertainty is on the level of attribute values.

Our approach stems from the idea of a probabilistic relational data model proposed by Barbará, Garcia-Molina and Porter [3]. By assuming keys are definite, the probability of an attribute value is a conditional probability depending on the key value of that tuple. To illustrate, consider the following relation, where *name* is the key attribute.

name	*city*	*specialty*	*age*
Jesse	[Taipei$^{0.4}$, Hsinchu$^{0.5}$, Kaohsiung$^{0.1}$]	SE	30
Annie	Kaohsiung	[DB$^{0.2}$, AI$^{0.8}$]	27

This relation describes two entities, "Jesse" and "Annie". The probability that Jesse's city is Taipei is

$$Prob(city = \text{"Taipei"} \mid name = \text{"Jesse"}) = 0.4$$

Now we define a probabilistic partial value as follows.

DEFINITION **4.1** A *probabilistic partial value*, denoted $\xi = [a_1^{p_1}, a_2^{p_2}, \ldots, a_n^{p_n}]$, associates with n possible values, $a_1, a_2, \ldots, a_n$, of the same domain $D \cup \{*\}$, where each a_i associates with a probability $p_i \neq 0$ such that $\sum_{i=1}^{n} p_i = 1$.

A relation consisting of tuples with probabilistic partial values is called a *probabilistic partial relation*. In general, probabilities can be assigned according to the timeliness of data, statistical information, etc. In [37][38], we present an application to show how to assign the probabilities of a partial value in heterogeneous database systems.

The equality of probabilistic partial values is defined as follows.

DEFINITION **4.2** Two probabilistic partial values, say $\xi_a = [a_1^{p_1}, a_2^{p_2}, \ldots, a_n^{p_n}]$ and $\xi_b = [b_1^{q_1}, b_2^{q_2}, \ldots, b_n^{q_n}]$, are said to be *equal*, denoted $\xi_a = \xi_b$, if $a_i = b_i$ and $p_i = q_i$, for each $i = 1, \ldots, n$.

Notice that, however, two *proper partial values* can never be equal (refer to Section 3.1).

We also use ν to denote the function that maps a probabilistic partial value to its corresponding finite set of possible values. That is, for a probabilistic partial value $\xi = [b_1^{p_1}, b_2^{p_2}, \ldots, b_n^{p_n}]$, $\nu(\xi) = \{b_1^{p_1}, b_2^{p_2}, \ldots, b_n^{p_n}\}$. Besides, a definite value d can be regarded as a probabilistic partial value $[d^1]$, and vice versa. In the following, $\nu(\xi)$ and ξ are used interchangeably when it does not cause confusion.

We define the cartesian product of two probabilistic partial values as follows.

DEFINITION **4.3** The *cartesian product* $\xi_a \stackrel{\sim}{\times} \xi_b$ of the probabilistic partial values $\xi_a = [a_1^{p_1}, a_2^{p_2}, \ldots, a_m^{p_m}]$ and $\xi_b = [b_1^{q_1}, b_2^{q_2}, \ldots, b_n^{q_n}]$ is the probabilistic partial value $\xi_{a \stackrel{\sim}{\times} b}$ with $\nu(\xi_{a \stackrel{\sim}{\times} b})$ being a set of the ordered pairs (a_i, b_j) with probability $p_i \times q_j$ for every $a_i^{p_i} \in \xi_a$ and $b_j^{q_j} \in \xi_b$.

By assuming non-key attributes of a relation to be independent, a tuple of a relation $r(A_1, A_2, \ldots, A_m)$, $\boxed{\xi_1, \xi_2, \ldots, \xi_m}$, can be transformed into the semantically-equivalent probabilistic partial value, $\xi_1 \stackrel{\sim}{\times} \xi_2 \stackrel{\sim}{\times} \ldots \stackrel{\sim}{\times} \xi_m$.

EXAMPLE **4.1** Consider the following relation $r(A_1, A_2)$

r

A_1	A_2
1	$[a^{0.2}, b^{0.8}]$
$[2^{0.3}, 3^{0.7}]$	c

We can regard $\{A_1, A_2\}$ as a composite attribute A_1A_2 and transform r in the following:

r

A_1A_2
$[(1,a)^{1\times 0.2}, (1,b)^{1\times 0.8}]$
$[(2,c)^{0.3\times 1}, (3,c)^{0.7\times 1}]$

$=$

r

A_1A_2
$[(1,a)^{0.2}, (1,b)^{0.8}]$
$[(2,c)^{0.3}, (3,c)^{0.7}]$

□

That is, a probabilistic partial relation $R(A_1, A_2, \ldots, A_m)$ can be regarded as a relation $R(A_1A_2 \cdots A_m)$ with the composite attribute $A_1A_2 \cdots A_m$. Therefore, a probabilistic partial relation can be considered as a set of probabilistic partial values. By this, we have the following definitions.

DEFINITION **4.4** An *interpretation*, $\beta = (b_1^{p_1}, b_2^{p_2}, \ldots, b_m^{p_m})$, of a set of probabilistic partial values, $\Psi = \{\xi_1, \xi_2, \ldots, \xi_m\}$, is an assignment of values from Ψ such that $b_i^{p_i} \in \xi_i, 1 \leq i \leq m$.

By Definition 4.4, for a set of partial values $\Psi = \{\xi_1, \xi_2, \ldots, \xi_m\}$, $\nu(\xi_1) \times \nu(\xi_2) \times \cdots \times \nu(\xi_m)$ is the set of all interpretations of Ψ.

DEFINITION **4.5** For an interpretation $\beta = (b_1^{p_1}, b_2^{p_2}, \ldots, b_m^{p_m})$ of a set of probabilistic partial values $\Psi = \{\xi_1, \xi_2, \ldots, \xi_m\}$, the *probabilistic value set* of β is denoted $V_\beta = \bigcup_{1 \leq i \leq m} \{b_i\}$ with probability $p_\beta = \prod_{1 \leq i \leq m} p_i$.

For simplicity, we use $V_\beta^{p_\beta}$ to represent both a probabilistic value set V_β and its probability p_β.

DEFINITION **4.6** For all interpretations, β_j, $1 \leq j \leq q$, $q = \mid \xi_1 \mid \times \mid \xi_2 \mid \times \cdots \times \mid \xi_m \mid$, of a set of probabilistic partial values $\Psi = \{\xi_1, \xi_2, \ldots, \xi_m\}$, the *family of probabilistic value sets* of Ψ is denoted $\mathcal{G}(\Psi) = \bigcup_{1 \leq j \leq q} \{V_{\beta_j}\}$, where V_{β_j} has the probability $p_{\beta_j} = \sum_{(\forall k)(V_{\beta_k} = V_{\beta_j})} p_{\beta_k}$. If $\Psi = \emptyset$ then define $\mathcal{G}(\Psi) = \emptyset$.

EXAMPLE **4.2** Consider the following relation $r(X, Y)$.

r

X	Y
a	y
a	$[x^{0.3}, y^{0.7}]$

We can regard $\{X, Y\}$ as a composite attribute XY and compute r to be

r

XY
$[(a,y)^1]$
$[(a,x)^{0.3}, (a,y)^{0.7}]$

Then there are two interpretations, β_1 and β_2, of r:

$$\beta_1 = ((a,y)^1, (a,x)^{0.3}) \quad \text{and} \quad \beta_2 = ((a,y)^1, (a,y)^{0.7}).$$

Their corresponding value sets are

$$V_{\beta_1} = \{(a,y), (a,x)\} \quad \text{and} \quad V_{\beta_2} = \{(a,y)\}.$$

with the corresponding probabilities

$$p_{\beta_1} = 1 \times 0.3 = 0.3 \quad \text{and} \quad p_{\beta_2} = 1 \times 0.7 = 0.7.$$

Therefore, the family of value sets of r is

$$\begin{aligned} \mathcal{G}(r) &= V_{\beta_1}^{p_{\beta_1}} \cup V_{\beta_2}^{p_{\beta_2}} \\ &= \{\{(a,y), (a,x)\}^{0.3}, \{(a,y)\}^{0.7}\}, \end{aligned}$$

which corresponds to the following two relations:

$V_{\beta_1}^{0.3}$

X	Y
a	y
a	x

$V_{\beta_2}^{0.7}$

X	Y
a	y

4.2 The Extended Division for Handling Probabilistic Partial Values

In this section, we define the extended division for probabilistic partial values, denoted $\tilde{\div}$, as follows. We assume the divisor of an extended division contains definite values only.

Let the operands of an extended division $\tilde{\div}$ be $r(X,Y)$ and $s(Y)$, where X and Y are possibly composite. The result of $r \tilde{\div} s$ is defined as follows.

$$\{t^p \mid (t \in \bigcup_{V_{\beta_i} \in \mathcal{G}(r)} (V_{\beta_i} \div s)) \wedge (p = \sum_{(\forall V_{\beta_i}^{p_{\beta_i}} \in \mathcal{G}(r))(t \in (V_{\beta_i} \div s))} p_{\beta_i})\}.$$

The following example illustrates this.

EXAMPLE **4.3** For the relations r and s given below,

r

X	Y
a	$[x^{\frac{1}{3}}, y^{\frac{2}{3}}]$
b	y
a	$[x^{\frac{1}{4}}, y^{\frac{3}{4}}]$

s

Y
x
y

there are four interpretations,

$$\begin{aligned}
\beta_1 &= ((a,x)^{\frac{1}{3}}, (b,y)^1, (a,x)^{\frac{1}{4}}),\\
\beta_2 &= ((a,x)^{\frac{1}{3}}, (b,y)^1, (a,y)^{\frac{3}{4}}),\\
\beta_3 &= ((a,y)^{\frac{2}{3}}, (b,y)^1, (a,x)^{\frac{1}{4}}), \text{and}\\
\beta_4 &= ((a,y)^{\frac{2}{3}}, (b,y)^1, (a,y)^{\frac{3}{4}}).
\end{aligned}$$

The corresponding probabilistic value sets are shown below.

$V_{\beta_1}^{\frac{1}{12}}$

X	Y
a	x
b	y

$V_{\beta_2}^{\frac{1}{4}}$

X	Y
a	x
b	y
a	y

$V_{\beta_3}^{\frac{1}{6}}$

X	Y
a	y
b	y
a	x

$V_{\beta_4}^{\frac{1}{2}}$

X	Y
a	y
b	y

By these probabilistic value sets, we have

$$V_{\beta_1}^{\frac{1}{12}} \div s = \emptyset, \quad V_{\beta_2}^{\frac{1}{4}} \div s = \{a\}, \quad V_{\beta_3}^{\frac{1}{6}} \div s = \{a\}, \quad \text{and} \quad V_{\beta_4}^{\frac{1}{2}} \div s = \emptyset.$$

Therefore, the result $q = r \tilde{\div} s$ is

q

X	*probability*
a	$\sum\{\frac{1}{4}, \frac{1}{6}\} = \frac{5}{12}$

□

To compute the probability of an answer tuple of $r(X,Y) \stackrel{\sim}{\div} s(Y)$, all interpretations of r should be generated and then their corresponding probabilistic value sets be computed. However, this can be very time-consuming — in the worst case, its time complexity is exponential.

5 Discussion and Conclusion

Understanding and accommodating various types of uncertain data in databases have been an active research area. Moreover, the division operation is rarely discussed, especially when a database contains uncertain data. However, to process a query involving a universal quantifier, it is usually necessary to perform the division operation.

In this paper, we study the extended divisions for partial values and probabilistic partial values. The concept of probabilistic partial values is a generalization from that of partial values, which is in turn a generalization from that of null values. Due to the uncertainties of partial values and probabilistic partial values, the extended divisions may produce *maybe tuples* and *maybe tuples with probabilities*.

By employing the bipartite graph matching technique, we have shown the maybe result of an extended division for partial values can be obtained efficiently. We have also discussed the refinement on the maybe result to produce a more informative answer. However, as we have shown, the refinement process may corresponds to some NP-hard problem and therefore we have the trade-off between a more informative answer and the computational efficiency on the extended division operation over partial values.

Since maybe tuples cannot be compared to distinguish their degrees of uncertainty, maybe tuples with probabilities is more informative than that just with maybe status. However, it is time-consuming to derive the answer of an extended division that handles probabilistic partial values. Therefore, it is a trade-off between an efficient algorithm and a more informative answer. In the near future, we will investigate the parallelism of processing an extended division which handles probabilistic partial values, and develop an efficient parallel algorithm to speed up the computation.

References

[1] S. Abiteboul and G. Grahne, "Update Semantics for Incomplete Information", *Proc. 11th Int. Conf. Very Large Data Bases,* Stockholm, 1985, pp.1-12.

[2] F. Bancilhon and N. Spyratos, "Update Semantics of Relational Views", *ACM Trans. Database Systems*, Vol.6, No.4, 1981, pp.557-575.

[3] D. Barbará, H. Garcia-Molina, and D. Porter, "A Probabilistic Relational Data Model", *Lecture Notes in Computer Science: Advances in Database Technology — EDBT'90* (Springer-Verlag, NY, 1990), pp.60-74.

[4] P.A. Bernstein and D.M.W. Chiu, "Using Semi-Joins to Solve Relational Queries", *Journal of the Association for Computing Machinery,* Vol.28, No.1, 1981, pp.25-40.

[5] J. Biskup, "A Foundation of Codd's Relational Maybe-Operations", *ACM Trans. Database Systems*, Vol.8, No.4, 1983, pp.608-636.

[6] B. Bollobás, *Graph Theory: An Introductory Course* (Springer-Verlag, NY, 1979).

[7] J.A. Bondy and U.S.R. Murty, *Graph Theory with Applications,* (Macmillan Press, NY, 1976).

[8] A.L.P. Chen, J.S. Chiu, and F.S.C. Tseng, "Evaluating Aggregate Operations over Imprecise Data", *IEEE Transactions on Knowledge and Data Engineering* (to appear).

[9] J.S. Chiu and A.L.P. Chen, "A Sound and Complete Extended Relational Algebra for Exclusive Disjunctive Data", in *VLDB'94 Poster Paper Collection* (1994).

[10] J.S. Chiu and A.L.P. Chen, "An Exploration of Relationships among Exclusive Disjunctive Data", *IEEE Transactions on Knowledge and Data Engineering* (to appear).

[11] J.S. Chiu and A.L.P. Chen, "A Note on Incomplete Relational Database Models Based on Intervals", *IEEE Transactions on Knowledge and Data Engineering* (to appear).

[12] E.F. Codd, "Extending the Database Relational Model to Capture More Meaning", *ACM Trans. Database Systems*, Vol.4, No.4, 1979, pp.397-434.

[13] E.F. Codd, "Missing Information (Applicable and Inapplicable) in Relational Databases", *SIGMOD RECORD*, Vol.15, No.4, 1986, pp.53-78.

[14] E.F. Codd, "More Commentary on Missing Information in Relational Databases (Applicable and Inapplicable Information)", *SIGMOD RECORD*, Vol.16, No.1, 1987, pp.42-50.

[15] C.J. Date, *An Introduction to Database Systems* (Addsion-Wesley, MA, 5th ed., 1990).

[16] L.G. DeMichiel, "Resolving Database Incompatibility: An Approach to Performing Relational Operations over Mismatched Domains", *IEEE Trans. Knowledge and Data Engineering*, Vol.1, No.4, 1989, pp.485-493.

[17] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (Freeman, SF, 1979).

[18] J. Grant, "Null Values in a Relational Data Base", *Information Processing Letters*, Vol.6, No.5, 1977, pp.156-157.

[19] J. Grant, "Partial Values in a Tabular Database Model", *Information Processing Letters*, Vol.9, No.2, 1979, pp.97-99.

[20] J.E. Hopcroft and R.M. Karp, "An $n^{5/2}$ Algorithm for Maximum Matching in Bipartite Graphs", *SIAM J. Computing*, Vol.2, No.4, 1973, pp.225-231.

[21] T. Imielinski and W. Lipski, "On Representing Incomplete Information in a Relational Database", *Proc. 7th Int. Conf. Very Large Data Bases,* 1981, pp.389-397.

[22] T. Imielinski and W. Lipski, "Incomplete Information and Dependencies in Relational Databases", *Proc. ACM SIGMOD Int. Conf. Management of Data,* 1983, pp.178-184.

[23] E. Lien, "Multivalued Dependencies with Null Values in Relational Databases", *Proc. 5th Int. Conf. Very Large Data Bases,* 1979, pp.61-66.

[24] W. Lipski, "On Semantic Issues Connected with Incomplete Information Systems", *ACM Trans. Database Systems*, Vol.4, No.3, 1979, pp.262-296.

[25] K.-C. Liu and R. Sunderraman, "Indefinite and Maybe Information in Relational Databases", *ACM Trans. Database Systems*, Vol.15, No.1, 1990, pp.1-39.

[26] K.-C. Liu and R. Sunderraman, "A Generalized Relational Model for Indefinite and Maybe Information", *IEEE Trans. Knowledge and Data Engineering*, Vol.3, No.1, 1991, pp.65-77.

[27] D. Maier, *The Theory of Relational Databases*, (Computer Science Press, Rockville, MD, 1983).

[28] P.L. Meyer, *Introductory Probability and Statistical Applications* (Addison-Wesley, 2nd Edition, 1970).

[29] A. Motro, "Accommodating Imprecision in Database Systems: Issues and Solutions", *ACM SIGMOD RECORD*, Vol.19, No.4, 1990, pp.69-74.

[30] M.A. Roth and H.F. Korth, "The Design of ¬1NF Relational Databases into Nested Normal Form", *Proc. ACM SIGMOD Int. Conf. Management of Data,* 1987, pp.143-159.

[31] F. Sadri, "Modeling Uncertainty in Databases", *Proc. 7th IEEE Int. Conf. Data Engineering*, 1991, pp.122-131.

[32] F. Sadri, "Reliability of Answers to Queries in Relational Databases", *IEEE Trans. Knowledge and Data Engineering,* Vol.3, No.2, 1991, pp.245-251.

[33] P.S.M. Tsai and A.L.P. Chen, "Querying Uncertain Data in Heterogeneous Databases", in *IEEE 3rd Int'l Workshop on Research Issues in Data Engineering: Interoperability in Multidatabase Systems (RIDE-IMS)* (1993).

[34] P.S.M. Tsai and A.L.P. Chen, "A Localized Approach to Query Optimization in Heterogeneous Database Systems", *Information Science and Engineering*, 1993, pp. 605-623.

[35] F.S.C. Tseng, A.L.P. Chen, and W.P. Yang, "On Mapping Natural Language Constructs into Relational Algebra through E-R Representation", *Data & Knowledge Engineering*, Vol.9, No.1, 1992, pp.97-118.

[36] F.S.C. Tseng, A.L.P. Chen, and W.P. Yang, "Searching a Minimal Semantically-Equivalent Subset of a Set of Partial Values", *the International Journal of Very Large Data Bases—the VLDB Journal*, October 1993, pp.489-512.

[37] F.S.C. Tseng, A.L.P. Chen, and W.P. Yang, "A Probabilistic Approach to Query Processing in Heterogeneous Database Systems", in *IEEE 2nd Int'l Workshop on Research Issues in Data Engineering: Transaction and Query Processing (RIDE-TQP)* (1992) 176-183.

[38] F.S.C. Tseng, A.L.P. Chen, and W.P. Yang, "Answering Heterogeneous Database Queries with Degrees of Uncertainty", *Distributed and Parallel Databases: An International Journal*, 1993, pp. 281-302.

[39] Y. Vassiliou, "Null Values in Data Base Management — A Denotational Semantics Approach", *Proc. ACM-SIGMOD Int. Conf. Management of Data,* 1979, pp.162-169.

[40] Y. Vassiliou, "Functional Dependencies and Incomplete Information", *Proc. 6th Int. Conf. Very Large Data Bases,* 1980, pp.260-269.