An Exploration of Relationships Among Exclusive Disjunctive Data

Jui-Shang Chiu and Arbee L.P. Chen, *Member, IEEE Computer Society*

***Abstract*—In this paper, we elaborate on how to interpret the query answer on exclusive disjunctive databases and how to reduce the query answer into a more concise form. Exclusive disjunctive data are represented as a pair of value set and variable set in *Pv-table* which is an extension of the relational model. A value set corresponds to a finite set of *possible* values in which exactly one value is the *true* value. By variable sets, tuples may be related with certain relationships, namely *disjunctive relationship* and *join relationship*. Three kinds of tuple sets are classified according to these relationships, each possesses an important property, namely *co-exist, co-nonempty*, or *co-instance*. Based on these properties, the interpretation of Pv-tables can be formalized in a semantically meaningful way. Also, the redundant and mergeable tuples can be identified. After removing and merging tuples accordingly, a more concise Pv-table can thus provide a better understanding of the query result.**

***Index Terms*—Incomplete information, disjunctive information, partial values, query language semantics, tuple relationships, relational databases.**

I. Introduction

INCOMPLETE information in relational databases has been extensively studied. Different kinds of incomplete information that have been studied include null values [2], [4], [9], [10], [11], [20], [39], [43], [48], partial values [12], [16], [17], [40], [41], indefinite information [18], [22], [23], [32], [46], and maybe information [28], [29], [30], [33], [34], [35]. A *null value* represents a value unknown at present. A *partial value* represents a finite set of possible values in which exactly one value is the true value. More generally, *disjunctive information* corresponds to a finite disjunction of formulas, which can be a disjunction of attribute values or a disjunction of tuples. Disjunctive information is *indefinite* if at least one of the formulas must be true in the real world, and *maybe* otherwise. Disjunctive information is *exclusive* if only one of the formulas can be true, and *inclusive* if more than one of the formulas can be true. In this paper, we focus our attention on exclusive disjunctive information (which can be indefinite or maybe).

In the following we demonstrate two of the common approaches in representing and manipulating disjunctive information. Some of their limitations will be noted. The two approaches are tables with marked partial values [12], [35] and C-tables [20]. For illustration, the same relation "Student" with disjunctive information is represented by these two approaches as shown in Table I. The partial values are *marked* by variables to identify shared incomplete information, e.g., students Mary and Susan (or Susanna) have the same major. An extra attribute is used to indicate the status of a tuple, which can be "true" (representing either definite or indefinite information), or "maybe" (representing maybe information). *C-table* extends the relational model by adding a "Condition" attribute which contains a formula for each tuple. If the formula is other than "true," the information is assumed maybe. Notice that indefinite information cannot be clearly distinguished from maybe information in C-table.

TABLE I
AN EXAMPLE OF RELATION "STUDENT" WITH DISJUNCIVE INFORMATION

Student:

Name	Major	Status
John	CS	*true*
Mary	$\{CS, EE\}_x$	*true*
Paul	$\{Math, EE\}_y$	*true*
$\{Susan, Susanna\}_u$	$\{CS, EE\}_x$	*true*

Name	Major	Condition
John	CS	*true*
Mary	x	$(x = CS \vee x = EE)$
Paul	y	$(y = Math \vee y = EE)$
u	x	$(u = Susan \vee u = Susanna) \wedge (x = CS \vee x = EE)$

The information in a table (i.e., extended relation) is interpreted by mapping variables to values [38]. For example, student Mary majors in CS if variable x is mapped to CS, and EE if x is mapped to EE. Each interpretation of a table results in one possible relation. A table with disjunctive information represents a set of possible relations each comes from a unique interpretation.

TABLE II
THE RESULT OF SELECTION ON STUDENT RELATION

Student:

Name	Major	Status
John	CS	*true*
Mary	$\{CS\}_x$	*maybe*
$\{Susan, Susanna\}_u$	$\{CS\}_x$	*maybe*

Name	Major	Condition
John	CS	*true*
Mary	x	$(x = CS)$
u	x	$(u = Susan \vee u = Susanna) \wedge (x = CS)$

Now consider the query: find all the students who major in CS. John qualifies as a definite answer while Mary and Susan (or Susanna) qualify as maybe answers. The query results represented by marked partial values and formulas in C-table are shown in Table II. In the former, each unqualified major is

Manuscript received June 22, 1993; revised Apr. 29, 1994.
J.S. Chiu and A.L.P. Chen are with the Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan 30043, Republic of China; e-mail: alpchen@cs.nthu.edu.tw.
To order reprints of this article, e-mail: transactions@computer.org, and reference IEEECS Log Number K95081.

removed and the status is changed into "maybe." Since the status is of tuple-level, there is no way to identify which attribute contributes to the maybe information. In the latter, the formula (x = CS $\vee$ x = EE) is subsumed (and replaced) by the formula (x = CS). In both cases, the information about the possibility that Mary majors in EE is lost.

Moreover, consider another query: find all pairs of students who have the same major. The join result of students Mary and Paul is represented by marked partial values as <Mary, Paul, $\{\text{EE}\}_z$>. In order to correctly interpret the query result, if variable z is mapped to EE, x and y must be mapped to EE too. However, the relationship between x, y, and z is not specified. This may result in incorrect interpretations of answers.

In [8], we proposed an extended relational model called *Pv-table* to represent exclusive disjunctive information by a pair of variable set and value set. Pv-table is different from other existing extended models with partial values in the following two aspects (to be detailed in Section III):

1) The values which do not satisfy the predicates are retained and *signed* as unqualified. For example, the value of Major for Mary in the result of the first query will be represented by $(\{x\}, \{\text{CS}\,\overline{\text{EE}}\})$. By this representation, the information about both the data and the query can be preserved. Also, the attribute which contributes to maybe information can be identified.
2) Each value set is coupled with a variable set instead of a single variable. In evaluating a join operation, all variables in the two variable sets are copied to the resultant variable set. For example, joining the tuples of students Mary and Paul on Major will result in $(\{x, y\}, \{\text{EE}\,\overline{\text{CS}}\,\overline{\text{Math}}\})$ as the value of Major. Hence, the relationship between the join result and the original values is kept.

By (1) and (2) together, we have shown in [8] that queries on Pv-tables can be evaluated in a semantically correct manner.

In this paper, we elaborate on how to interpret the query answer on exclusive disjunctive databases (more specifically, Pv-tables) and how to reduce the query answer into a more concise form. A query result may be separated into definite, indefinite and maybe answers. It is not difficult to identify definite and maybe answers from the query result. However, it is not easy to identify indefinite answers. An indefinite answer may be represented by either a single tuple (with disjunctive information) or a disjunction of tuples. These tuples are related with certain relationships by variables. The difficulty of identifying indefinite answers stems from complicate relationships among tuples. Hence, it is crucial to explore the relationships among tuples first before we can correctly and efficiently interpret query answers from disjunctive databases.

Users may also be concerned about which type of answers a candidate tuple represents. For example, in the above selection, <John, CS> is a definite answer while <Mary, CS> is a maybe answer. A definite answer appears in every possible relation while a maybe answer appears only in some of the possible relations. An indefinite (exclusive) answer is a disjunction of tuples in which exactly one tuple is in each possible relation. By examining how the candidate tuples appear in the set of possible relations, the type of answers they represent can be determined. However, since the number of possible relations can be enormous, it would be very inefficient to examine all the possible relations. Instead, the efficiency will be much improved if we only examine their corresponding tuples in Pv-table. The problem then becomes to find out such corresponding tuples and to know the relationships among these tuples.

There are two kinds of relationships among tuples, namely *disjunctive relationship* and *join relationship*. Due to its power to express these relationships in a Pv-table, good properties are preserved in this model [8]. We classify three kinds of tuple sets based on these relationships, each possesses an important property, namely *co-exist, co-nonempty*, or *co-instance*. We shall show that the types of information a set of tuples represent can be determined efficiently according to the properties they possess. Moreover, we shall show that the interpretation of Pv-tables can be formalized in a semantically meaningful way.

Query answers may contain redundancies. The major sources of redundancies come from projection and union operations. Also, under certain circumstances, some of the tuples in query answers can be merged. A Pv-table would be more concise if the redundant tuples are identified and removed and the mergeable tuples are merged. We shall show how the redundant and mergeable tuples can be identified by examining the relationships among tuples and show how they are merged. Two kinds of the reduction process will be discussed separately according to whether they are commutative with relational operations.

The rest of this paper is organized as follows: Section II presents related work. In Section III, we introduce the basic concept of Pv-tables. In Section IV, we show how to interpret information from Pv-tables. Three kinds of properties for tuple sets are classified according to the relationships among tuples. The interpretation of Pv-tables can then be formalized based on these properties. Section V discusses how to reduce a Pv-table in a more concise form by removing redundant tuples and by merging tuples. Section VI concludes our work. In order to make this paper more readable, we present the proofs to all theorems and lemmas in the Appendix.

II. Related Work

This section reviews related approaches to the problem of representing and manipulating incomplete information.

The work on incomplete information is pioneered by Codd [9], [10] on extending relational algebra to manipulate null values. Lipski [28] provides two different interpretations of queries, the internal and the external. The internal interpretation ignores all incomplete information, referring only to what is known to the system. In contrast, the external interpretation refers to the real world modeled by the system with incomplete information. The external interpretation has two bounds: The lower bound $\|Q\|_l$ corresponds to the definite information; and the upper bound $\|Q\|_h$ corresponds to the union of definite and maybe information. Indefinite information was not distinguished from maybe information in this interpretation.

Imielinski and Lipski [20] examine the expressive power of three extended relational models with null values based on a semantic correctness criterion (also see [31]). Let rep(T) denote the set of possible relations represented by table T. The correctness criterion states that for each operator f, the extended operator f_T on T should be defined to satisfy the following conditions:

1) $rep(f_T(T)) = f_\Sigma(rep(T))$ for unary f and
2) $rep(f_T(T_1, T_2)) = f_\Sigma(rep(T_1), rep(T_2))$ for binary f

where $f_\Sigma(rep(T))$ is defined as $\{f(R) \mid R \in rep(T)\}$ and $f_\Sigma(rep(T_1), rep(T_2))$ as $\{f(R_1, R_2) \mid R_1 \in rep(T_1) \text{ and } R_2 \in rep(T_2)\}$.

The three models examined are *Codd-table*, a table with the usual Codd's null values, *V-table*, a table with marked null values, and *C-table*, a table with marked null values and logical formulas. The result indicates that only C-table can be evaluated in a semantically correct manner with respect to the primitive relational operators (namely, selection, projection, Cartesian product, intersection, join, union, and difference). However, C-tables do not distinguish between indefinite and maybe information.

Grant [16], [17] extends null value to partial value. DeMichiel [12] extends the relational model with partial values to resolve the domain mismatch problem in heterogeneous database systems. Tseng et al. [40] further extend partial values with probabilities for answering heterogeneous database queries. Ola and Ozsoyoglu [35] use marked partial values to identify shared incomplete information. Liu and Sunderraman [29], [30] consider inclusive disjunctions instead of exclusive disjunctions. Under all these models, maybe answers for some types of queries may actually be indefinite answers.

Grant and Minker [18], [32] consider query answering for indefinite (disjunctive) databases that contain disjunctive formulas in first-order logic. A disjunctive formula corresponds to a set of possible tuples under the relational model. An algorithm to find definite and indefinite answers to a query was developed. Yuan and Chiang [46] develop a sound and complete query evaluation algorithm for relational databases with disjunctive information. Their algorithm proceeds by recursively decomposing complex queries in a normal form into extended relational operations on simpler queries. The algorithm returns all definite and indefinite answers to a given query. However, how to represent and manipulate maybe answers in the intermediate steps of the evaluation process was not discussed. Ignoring maybe answers is one of the sources of the incompleteness of subsequent query evaluations.

In [8], we propose Pv-table to represent definite, indefinite as well as maybe information. We showed that query evaluation on Pv-tables is sound and complete for queries consisting of extended selection, union, intersection, Cartesian product and join. The query evaluation on Pv-tables is based on set operations instead of logic operations. The complexity of the evaluation is lower than on C-tables, but higher than other extended models with partial values.

A classification of extended relational models with incomplete information is given in Fig. 1.

Further related work is described in the following. Levesque [25] concentrates on the formal aspects on incomplete information in knowledge-based systems. His approach considers not only the issues of handling incomplete information about data, but also how to access the knowledge about the incompleteness. Levene and Loizou [24] define the semantics of nested relations with null values in terms of integrity constraints. Imielinski et al. [21] were motivated by the applications within design, planning and scheduling areas. The object-oriented data model was extended with an OR-type whose instances are OR-objects. OR-objects are, in fact, marked partial values with variables representing object identifiers. In [22], Imielinski and Vadaparty give a complete syntactic characterization of CoNP-Complete [14] conjunctive queries for databases with OR-objects. Libkin and Wong [26] investigate the relationship between query languages and normalization. Losslessness of normalization was established for a large class of queries.

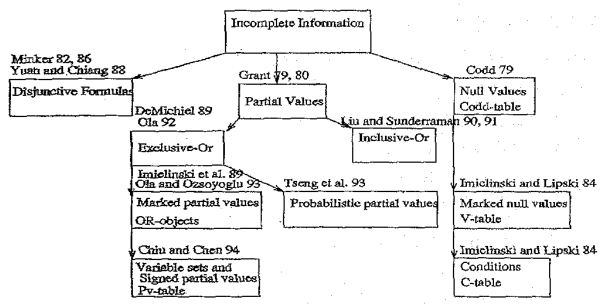


Fig. 1. A classification of extended relational models with incomplete information.

In addition to the work on query processing, the issues of dependency satisfaction and updating of incomplete information have also been studied. The former work includes [15], [19], [27], [42], [44] and the latter includes [1], [45].

Incomplete information can be classified into two aspects, i.e., imprecise information and uncertain information. Dubois and Prade [13] distinguish between imprecise and uncertain information by stating that the concept of *imprecision* is relevant to the content of an attribute value, while the concept of *uncertainty* is relevant to the degree of truth of its attribute value. In this paper, we focus our attention on the imprecise aspect. To the uncertain aspect, two major approaches are the possibility approach and the probability approach. The former approach is based on the fuzzy set theory which was first introduced by Zadeh [47]. Various kinds of fuzzy relational databases were proposed, such as, Buckles and Petry [5], Prade and Testemale [36], and Zemankova [49]. The work with the latter approach includes Barbará et al. [3], Cavallo and Pittarelli [6], and Tseng et al. [40].

III. Extended Relations for Exclusive Disjunctive Data

In this section, we give the definition of Pv-table in two steps. In the first step, we assume that a Pv-table contains only definite and indefinite information. In the second step, we consider a Pv-table with maybe information also.

NOTATION. Throughout this paper, we use U to denote a fixed, finite set of attributes. Attribute is denoted by A, set of attributes by X, tuple by t and relation by R with possible subscripts. Associated with each $A \in U$ is a value domain $D(A)$ and a variable domain $V(A)$.

DEFINITION 3.1. *A Pv-tuple t on X is a mapping that associates with every $A \in X$ a set couple, i.e., $t(A) = (v, \rho)$ where the variable set $v \subseteq V(A)$, and the value set $\rho \subseteq D(A)$, $\rho \neq \emptyset$. Such a set couple $t(A)$ is called a Pv-value. A Pv-table Tv on X is a finite set of Pv-tuples on X.*

NOTATION. For clarification, the Pv-value $t(A)$, its variable set and value set are denoted as $t.A$, $t.A.v$ and $t.A.\rho$, respectively. A Pv-tuple t is denoted as $<(t.A_1.v,\ t.A_1.\rho),\ \cdots,\ (t.A_n.v,\ t.A_n.\rho)\}$ where $A_i \in X$ for $1 \leq i \leq n$. A set of variables $\{u, v, \cdots, y, z\}$ is abbreviated as $u\ v \cdots y\ z$, and values $\{a, b, \cdots, e, f\}$ as $a\ b \cdots e\ f$.

The value set corresponds to a set of possible values in which exactly one value is the true value. Let $|\rho|$ denote the *cardinality* of a value set ρ. A value set is *definite* if its cardinality equals to 1. When a Pv-table is created, each value set is associated with a variable set. The variable set is empty if the value set is definite, otherwise it contains a single variable. Moreover, we assume that the information contained in Pv-tables is *consistent*. That is, if two Pv-values contain the same nonempty variable set, their initial value sets must be the same and they represent the same true value. All operations defined on Pv-tables should result in consistent Pv-tables.

EXAMPLE. A Pv-table representing the relation "Student" is shown in Table III. Since t_2.Major and t_4.Major are the same, by consistent assumption, we can tell that the two students are of the same major. Conversely, if a Pv-table contains, say, both (x, CS EE) and (x, CS Math), it is not consistent.

TABLE III
AN EXAMPLE OF PV-TABLE

Student:	Name	Major
t_1:	($\emptyset$, John)	($\emptyset$, CS)
t_2:	($\emptyset$, Mary)	(x, CS EE)
t_3:	($\emptyset$, Paul)	(y, Math EE)
t_4:	(u, Susan Susanna)	(x, CS EE)

Value sets may be restricted by predicates specified in relational operators such as selection, join, etc. Consequently, some of the values in value sets may become *unsatisfiable*. In the following, two ways to represent the unsatisfiable values are introduced. They are

1) to sign the value with a bar and
2) to replace the value by the symbol ψ.

We illustrate with several examples the query evaluation on Pv-tables which will result in maybe information. The definition of Pv-table will be extended with these two kinds of symbols to represent maybe information.

EXAMPLE. Consider a selection with predicate (Name = Susan or Major = CS) on Pv-table Student shown in Table III. For t_2, as EE is unsatisfiable, we have (x, CS$\overline{\text{EE}}$) in the result. From this result, we can tell that the major is either CS or EE but EE is unsatisfiable in the query. Hence, both semantics of the data and the query are preserved. As for t_3, since both Math and EE are unsatisfiable, the Pv-tuple itself is unsatisfiable and cannot be included in the result. The selection on t_4 will result in two Pv-tuples $<$(u, Susan $\overline{\text{Susanna}}$), ($x$, CS EE)$>$ and $<$(u, Susan Susanna), (x, CS$\overline{\text{EE}}$)$>$ corresponding to subpredicates (Name = Susan) and (Major = CS), respectively. Note that the two resulting Pv-tuples are regarded as consistent. Their initial value sets are the same though certain values are signed unsatisfiable for the corresponding subpredicate. The result of the selection on Pv-table Student is shown in Table IV.

TABLE IV
AN ILLUSTRATION OF SELECTION ON THE PV-TABLE

CS-Student:	Name	Major
	($\emptyset$, John)	($\emptyset$, CS)
	($\emptyset$, Mary)	(x, CS$\overline{\text{EE}}$)
	(u, Susan Susanna)	(x, CS EE)
	(u, Susan $\overline{\text{Susanna}}$)	(x, CS$\overline{\text{EE}}$)

After applying certain relational operations, some unsatisfiable values may be signed with bars or be replaced by ψ. Two partial values which contain the same nonempty variable set are regarded as consistent if one of the following two cases holds:

1) the value sets are the same while ignoring the bar signs;
2) one value set is a subset of the other value set and both value sets contain ψ.

The *logical formula* corresponding to a Pv-value, say, (x, CS$\overline{\text{EE}}$) is $(x = \text{CS} \vee x = \text{EE}) \wedge (x \neq \text{EE})$. The formula is logically equivalent to $(x = \text{CS})$. However, the semantics of the former is richer. The formula corresponding to a Pv-tuple is a conjunction of formulas corresponding to its Pv-values. Let t' be the resultant Pv-tuple by applying query predicates on a Pv-tuple t. All operations should be defined such that the evaluation of formula corresponding to t' is false iff the evaluation of predicates on t is false.

EXAMPLE. Consider the equi-join of two Pv-values, (x, CS EE) and ($\emptyset$, CS). It is easy to see that the join will succeed if x = CS and fail otherwise. As in the case of selection, EE is then replaced by $\overline{\text{EE}}$. The result of this join is represented as (x, CS$\overline{\text{EE}}$).

EXAMPLE. Consider another equi-join, (x, CS EE) with (z, CS Math). Clearly, $(x = z)$ will be false whenever $x \neq$ CS or $z \neq$ CS. That is, both EE and Math are unsatisfiable. The result of this join is represented as (xz, CS ψ) where ψ can be regarded as a set of values (e.g., EE and Math) which are unsatisfiable. Note that the two variable sets are unioned in the resultant Pv-value. The relationship between the resultant Pv-value and others are therefore retained. For a Pv-value to be satisfiable, the true value of all variables in the set must be identical.

In the case where neither Pv-value to be joined is definite, the unsatisfiable values are removed and ψ is included in the result. Moreover, once ψ is included in, say $t.A.\rho$, from then on, values in $t.A.\rho$ which are unsatisfiable will be removed.

EXAMPLE. Consider Pv-tables Tv_1, Tv_2, and Tv_3 on $\{A\}$ as shown in Table V. Tv_3 is the join result of Tv_1 and Tv_2. We shall show later that Tv_3 can be reduced to a more concise Pv-table. Tv_3 will be frequently referred throughout the rest of this paper.

TABLE V
AN ILLUSTRATION OF JOIN

Tv_1:

A
(x, ab)
(y, cd)

Tv_2:

A
(w, ac)
$(\varnothing, a)$
$(\varnothing, b)$
(z, bd)

Tv_3:

	A
t_1:	$(xw, a\psi)$
t_2:	$(yw, c\psi)$
t_3:	$(x, a\bar{b})$
t_4:	$(x, \bar{a}b)$
t_5:	$(zx, b\psi)$
t_6:	$(zy, d\psi)$

According to the above discussion, the definition of Pv-tuple is extended as follows.

DEFINITION 3.2. *Let $\overline{D}(A)$ be the set of $\bar{a}$ for every $a \in D(A)$, $D(A) \cap \overline{D}(A) = \varnothing$ for all $A \in U$. A Pv-tuple t on X is a mapping that associates with every $A \in X$ a set couple, i.e., $t(A) = (v, \rho)$ where the variable set $v \subseteq V(A)$, and the value set $\rho \subseteq D(A) \cup \overline{D}(A) \cup \{\psi\}$, $\rho \cap D(A) \neq \varnothing$, $a \in \rho \Rightarrow \bar{a} \notin \rho$, $\psi \in \rho \Rightarrow \rho \cap \overline{D}(A) = \varnothing$,* and $|\rho| = 1 \Rightarrow v = \varnothing$.

DEFINITION 3.3. *A Pv-value $t(A) = (v, \rho)$ is* definite *if $|\rho| = 1$,* indefinite *if $|\rho| > 1$ and $\rho \subseteq D(A)$, and* maybe *if $\rho \backslash D(A) \neq \varnothing$ where \ denotes set difference. A Pv-tuple t is definite if all its Pv-values are definite. t is indefinite if at least one of its Pv-values is indefinite and none is maybe. t is maybe if at least one of its Pv-values is maybe. A Pv-table Tv can be partitioned into three subsets Tv^D, Tv^I and Tv^M according to whether their tuple types are definite, indefinite and maybe, respectively.*

EXAMPLE. Pv-value $(\varnothing$, John) is definite and (u, Susan Susanna) is indefinite. Both (x, CS $\overline{\text{EE}}$) and (xy, CS ψ) are maybe.

Originally, a Pv-table contains only definite and indefinite Pv-tuples. To represent the query result in a Pv-table, Pv-tuples that possibly (but not surely) satisfy a query will be turned into maybe Pv-tuples. In the next section, we shall show that a set of maybe Pv-tuples which actually represent indefinite information can be identified. For the formal definitions of the extended relational operations, refer to [7].

IV. RELATIONSHIPS AND SEMANTIC INTERPRETATIONS

In this section, we show how to interpret information from Pv-tables in a semantically meaningful way. To interpret information from a Pv-table is to map it to relations. The *information content* of a Pv-table Tv, denoted by rep(Tv), is the set of all possible relations represented by Tv. Exactly one possible relation (which can be an empty set) in rep(Tv) is true in the real world. The relations not included in rep(Tv) is assumed to be false.

A. The Set of Possible Relations

Recall from Section III that variable sets are coupled with value sets in Pv-tables. In addition, a Pv-value corresponds to a logical formula and a Pv-tuple to a conjunction of logical formulas. Thus, the mapping from a Pv-table to a relation can be defined as a mapping from variables to values by which tuples whose formulas are satisfied are included in the relation.

In the following, we define the mapping of variables, logical formulas, Pv-tuples and Pv-tables. Then we give the definition of rep(Tv).

DEFINITION 4.1. *Let $V = \bigcup_{A \in U} V(A)$ and $D = \bigcup_{A \in U} D(A) \cup \{\psi\}$. A* valuation *is a mapping $\delta : V \to D$ such that $x \in V(A)$ implies $\delta(x) \in D(A) \cup \{\psi\}$.*

EXAMPLE. Consider Pv-table Student shown in Table III again. One of the mappings is to map u into Susan, x into CS, and y into Math. It is easy to see that there are totally eight meaningful mappings for Pv-table Student.

DEFINITION 4.2. *The logical formula of a Pv-tuple t is denoted by $\lambda(t)$. A valuation δ of $\lambda(t)$, $\delta(\lambda(t))$, is a logical formula in which each variable x in $\lambda(t)$ is replaced by $\delta(x)$.*

There are two cases which cause a formula to be false. One is that some of the variables are mapped into values which are unsatisfiable in a selection predicate. The other is that variables in the variable set are mapped into different values or ψ which are unsatisfiable in an equi-join. If neither case exists, the formula is true. Thus, we have the following lemma.

LEMMA 4.1. *Let t be a Pv-tuple in a Pv-table Tv on X. $\delta(\lambda(t))$ is true iff for every A in X there exists a value $a \in t.A.\rho \cap D(A) \wedge (\forall x \in t.A.v)(\delta(x) = a)$.*

EXAMPLE. Consider (xz, CS ψ), which is the result of joining (x, CS EE) and (z, CS Math). For the logical formula of (xz, CS ψ) to be true, both x and z must be mapped into CS.

DEFINITION 4.3. *A valuation δ of t, $\delta(t)$, is a relational tuple if $\delta(\lambda(t))$ is true, and undefined otherwise. The values of $\delta(t)$ is defined as $\delta(t).A = t.A.\rho$ if $t.A.v = \varnothing$, and $\delta(t).A = \delta(x)$, $x \in t.A.v$ otherwise.*

Let t_r be a relational tuple and t be a definite Pv-tuple. We said that t_r is *value equivalent* to t, denoted as $t_r \simeq t$, if $t_r = \delta(t)$.

DEFINITION 4.4. *A valuation δ of a Pv-table Tv maps Tv into a relation and is defined as $\delta(Tv) = \{\delta(t) \mid \delta(\lambda(t)) =$ true $\wedge\ t \in Tv\}$. $\delta(Tv)$ is an* empty relation *(i.e., an empty set) denoted $R_\varnothing$, if $\forall t \in Tv$, $\delta(t)$ is undefined.*

DEFINITION 4.5. *Assume that $A_i \neq A_j \Rightarrow V(A_i) \cap V(A_j) = \varnothing$. A valuation δ of a Pv-table Tv on X is* bounded *if for each $x \in t_i.A.v$, $t_i \in Tv$, $A \in X$, the following conditions hold:*

1) $\delta(x) \neq \psi \Rightarrow \delta(x) \in t_i.A.p \cap D(A) \cup \{a \mid \bar{a} \in t_i.A.p \cap \overline{D}(A)\}$,
2) $\delta(x) = \psi \Rightarrow (\psi \in t_i.A.\rho)$
   $\wedge\ (\nexists\, t_j \in Tv)(x \in t_j.A.v \wedge \psi \notin t_j.A.\rho)$.

Throughout the rest of this paper, when we mention a valuation, we mean a bounded valuation.

DEFINITION 4.6. rep(Tv), *the information content of Tv, is defined as* $\text{rep}(Tv) = \{R \mid (\exists\delta)(\delta(Tv) = R)\}$.

EXAMPLE. Consider Pv-table Tv_3 shown in Table V again. Let δ be defined as $\delta(w) = \delta(x) = a$, $\delta(y) = d$, and $\delta(z) = b$. We have $\delta(Tv_3) = \{<a>\}$. Consider another δ' with $\delta'(w) = c$, $\delta'(x) = b$, and $\delta'(y) = \delta'(z) = d$. We have $\delta'(Tv_3) = \{<b>, <d>\}$. The rest of mappings for Tv_3 can be derived similarly. rep(Tv_3) is shown in Table VI.

TABLE VI
THE SET OF ALL POSSIBLE RELATIONS

rep(Tv_3)

A	A	A	A	A	A
a	b	a	a	b	b
		c	d	c	d

It is worthwhile to emphasize here that indefinite information are distinguishable from maybe information in a Pv-table. Indefinite information is represented by either indefinite Pv-tuples or a disjunction of maybe Pv-tuples. If a Pv-table Tv contains such a disjunction $Tv_s \subseteq Tv^M$, $R_\varnothing \notin \text{rep}(Tv_s)$. The inclusion of empty relation in the definition of *rep* makes it semantically clearer.

B. Relationships Among Pv-Tuples

There are two kinds of relationships among Pv-tuples. One is called *disjunctive relationship* (*OR-ship*, for short), and the other *join relationship* (*joinship*, for short). According to these relationships, three kinds of tuple sets each possesses an important property will be classified in the next subsection.

DEFINITION 4.7. *Let Tv be a Pv-table on X. The condition of OR-ship is defined as*

$$\text{OR–ship}_{Tv}(t_i, t_j) = t_i \in Tv \wedge t_j \in Tv \wedge (\forall A \in X)\ (t_i.A.v = t_j.A.v).$$

If t_i and t_j are related with an OR-ship, it can be logically interpreted as $t_i \vee t_j$.

Recall that a selection with disjunctive predicates on a Pv-tuple will result in a set of Pv-tuples, each corresponds to a subpredicate. These Pv-tuples are related with an OR-ship. The union and projection operations may also result in Pv-tuples related with an OR-ship. Since they represent the same data, it is possible to unite them into one. We shall discuss how Pv-tuples can be united in Section V.A.

DEFINITION 4.8. *Let Tv be a Pv-table on X. The condition of joinship is defined as*

$$joinship_{Tv}(t_i, t_j) = t_i \in Tv \wedge t_j \in Tv \wedge (\exists A \in X)(t_i.A.v \cap t_j.A.v \neq \varnothing).$$

EXAMPLE. Let $t_s = <(x, ab)>$, $t_1 = t_s \rhd\!\lhd^* <(w, ac)> = <(xw, a\psi)>$, and $t_2 = t_s \rhd\!\lhd^* <(z, bd)> = <(xz, b\psi)>$. t_1 and t_2 are related with a joinship as they are joined by the same Pv-tuple t_s. Since joined variables are collected in variable sets, t_1 and t_2 share with variables in t_s (i.e., x). And, the truth of the formula of t_1 implies $x = w = a$ and that of t_2 implies $x = z = b$. Hence, t_1 and t_2 cannot both be true.

C. Co-exist, Co-nonempty, and Co-instance Sets

DEFINITION 4.9. *A set of Pv-tuples is* co-exist *if there exists a valuation which maps each Pv-tuple in the set to a relational tuple. The set is* co-nonempty *if there does not exist a valuation by which all the mappings of tuples in the set are undefined. It is* co-instance *if all valuations map it to at most one tuple. A set is* maximal co-exist *if it is co-exist and none of its proper superset is co-exist. It is* minimal co-nonempty *if it is co-nonempty and none of its proper subset is co-nonempty. Similarly, it is* maximal co-instance *if it is co-instance and none of its proper superset is co-instance.*

EXAMPLE. Consider Tv_3 in Table V again. Either t_3 or t_4 will be mapped into a tuple in any valuation. Thus, $\{t_3, t_4\}$ is not co-exist but co-nonempty. In contrast, it is not difficult to verify that $\{t_2, t_3\}$ is co-exist but not co-nonempty. The former is co-instance, but the latter is not.

Before discussing these three kinds of Pv-tuple sets, let us define some new notations.

Notation. Let $V_{Tv}(A) = \bigcup_{t\in Tv} t.A.v$ and $D_{Tv}(A) = \bigcup_{t\in Tv} t.A.\rho \cap D(A)$. Let $T_{Tv}(A, x)$ denote the set of Pv-tuples in Tv where the variable set of A contains variable x, i.e.,

$$T_{Tv}(A, x) = \{t \mid x \in t.A.v \wedge t \in Tv\}.$$

Note that Pv-tuples in $T_{Tv}(A, x)$ are related with a joinship on A by x.

EXAMPLE (cont'd). $V_{Tv_3}(A) = \{w, x, y, z\}$, $D_{Tv_3}(A) = \{a, b, c, d\}$, and $T_{Tv_3}(A, x) = \{t_1, t_3, t_4, t_5\}$.

C.1. The co-exist set

DEFINITION 4.10. *A valuation δ is a* concordant *valuation of Pv-table Tv if $\delta(t)$ is defined (i.e., $\delta(\lambda(t))$ is true), $\forall\ t \in Tv$.*

Note that a Pv-table Tv is co-exist iff there exists a concordant valuation for it. In the following, a set called *concordant closure* will be defined. We shall show that a concordant valuation will map all variables in a concordant closure into an identical value.

DEFINITION 4.11. *A* concordant closure *of x over A of Tv, denoted by $V^*_{Tv}(A, x)$, is a maximal subset of $V_{Tv}(A)$ holding the condition: $x \in V^*_{Tv}(A, x)$ and for every pair of variables in the set, say x_1 and x_m, there exists a set of variables $x_1, \ldots, x_i, x_{i+1}, \ldots, x_m$ such that $x_i \in t_i.A.v \cap t_{i+1}.A.v$,* for $1 \le i < m$. *$T^*_{Tv}(A, x)$ denotes a maximal subset* of *Tv where the variable set of A contains some variable(s) in $V^*_{Tv}(A, x)$, i.e.,*

$$T^*_{Tv}(A, x) = \{t \mid t.A.v \cap V^*_{Tv}(A, x) \neq \varnothing \wedge t \in Tv\}.$$

Let $\{t_1, \ldots, t_i, t_{i+1}, \ldots, t_m\}$ be the set $T^*_{Tv}(A, x)$ where $x_i \in t_i.A.v \cap t_{i+1}.A.v$. If $\delta(\lambda(t_{i+1}))$ is true, we have $\delta(x_i) = \delta(x_{i+1}) = a$ where $a \in t_i.A.\rho \cap t_{i+1}.A.\rho \cap D(A)$. If δ is concordant on Tv, we have $\delta(x_1) = \cdots = \delta(x_i) = \delta(x_{i+1}) = \cdots = \delta(x_m) = a$ where $a \in \bigcap_{t_i \in T^*_{Tv}(A,x)} t_i.A.\rho \cap D(A)$. That is, a concordant valuation will map all variables in a concordant closure into an identical value. Conversely, if a valuation δ maps all variables in a closure into an identical value for each concordant closure over Tv, by Lemma 4.1, δ is concordant. Therefore, the following lemma holds.

LEMMA 4.2. *Let Tv be a nonempty set of Pv-tuples on X. The condition of a co-exist set Tv is:*

$$co-exist(Tv) =$$
$$(\forall A \in X)(\forall x \in V_{TV}(A))\left(\bigcap_{t \in T^*_{TV}(A,x)} t.A.\rho \cap D(A) \neq \varnothing\right)$$

EXAMPLE (cont'd). $V^*_{\{t_2,t_3,t_4\}}(A,x) = \{x\}$. $T^*_{\{t_2,t_3,t_4\}}(A,x) = \{t_3,t_4\}$

Since $t_3.A.\rho \cap t_4.A.\rho \cap D(A) = \varnothing$, $\{t_2, t_3, t_4\}$ is not co-exist. However, $\{t_2, t_3\}$ is co-exist. Note that $t_2.A.v \cap t_3.A.v = \varnothing$. Thus, their variables are in different closures and can be mapped into different values. The maximal co-exist sets in Tv_3 are $\{t_1, t_3, t_6\}$, $\{t_2, t_3\}$, $\{t_2, t_4, t_5\}$, and $\{t_4, t_6\}$.

C.2. The co-nonempty set

DEFINITION 4.12. *A valuation δ is a* nonempty *valuation of Tv if $\delta(Tv) \neq R_\varnothing$.*

Note that Tv is co-nonempty if any valuation of Tv is nonempty. That is, $R_\varnothing \notin$ rep(Tv). It is clear that $R_\varnothing \notin$ rep(Tv^D) and $R_\varnothing \notin$ rep(Tv^I). Also, it is possible that Tv^M is co-nonempty. In the following, the nonempty valuation of a Pv-table with single attribute will be discussed first. The extension to the whole set of attributes then follows.

DEFINITION 4.13. *Let $Tv[A]$ denote the projection of Tv on A. Given that $Tv_s \subseteq Tv^M$, $Tv_s[A]$ is* co-indefinite *on x, if $Tv_s = T_{Tv_s}(A, x)$ and* rep($Tv_s[A]$) = rep($\{<(x, \rho)>\}$) *where ρ is a nonempty subset of $D(A)$ (which implies $R_\varnothing \notin$ rep($Tv_s[A]$)).*

EXAMPLE (cont'd). Consider t_3 and t_4 in Tv_3.

$$\text{rep}(\{<(x, a\bar{b})>, <(x, \bar{a}b)>\}) = \text{rep}(\{<(x, ab)>\}).$$

Hence, $\{t_3, t_4\}$ is co-indefinite on x. It means that the disjunction of t_3 and t_4 is indefinite.

Suppose that $Tv_s[A]$ is co-indefinite on x. The following conditions must hold:

1) $(\nexists\, t \in Tv_s[A])(\psi \in t.A.\rho)$. Suppose otherwise, $\psi \in t.A.\rho$ for some $t \in Tv_s[A]$. Let $\delta(x) = \psi$. We have $\delta(Tv_s[A]) = R_\varnothing$, a contradiction. Hence, $\nexists\, t \in Tv_s[A]$ such that $\psi \in t.A.\rho$. Also note that $\psi \notin t.A.\rho \Rightarrow |t.A.v| = 1$. Hence, $t.A.v = \{x\}$ for $t \in T_{Tv_s[A]}(A, x)$.
2) $\overline{D}(A) \cap \bigcap_{t \in Tv_s[A]} t.A.\rho = \varnothing$. Suppose otherwise, $\bar{a} \in \bigcap_{t \in Tv_s[A]} t.A.\rho \cap \overline{D}(A)$. Let $\delta(x) = a$. We have $\delta(Tv_s[A]) = R_\varnothing$, a contradiction too.

Based on the above discussion, we have the following lemma.

LEMMA 4.3. Let $Tv_s \subseteq Tv^M$, $Tv_s \neq \varnothing$, and $A \in X$. The condition of a co-indefinite set $Tv_s[A]$ is:

$$\text{co-indefinite}(Tv_s[A], x) = \left(Tv_s[A] = T_{Tv_s[A]}(A, x)\right) \wedge \left(\nexists\, t \in Tv_s[A]\right)(\psi \in t.A.\rho) \wedge \overline{D}(A) \cap \bigcap_{t \in Tv_s[A]} t.A.\rho = \varnothing.$$

LEMMA 4.4. *Let $Tv^a = \{t \mid a \in t.A.\rho \cap D(A) \wedge t \in Tv\}$. Let $Tv_s \subseteq Tv^M$, $Tv_s \neq \varnothing$ and $A \in X$. Tv^M is co-nonempty iff it contains a subset Tv_s satisfying the condition:*

$$\text{co-nonempty}(Tv_s)$$
$$= (\exists A \in X)\Big(\text{co-indefinite}(Tv_s[A], x) \wedge \left(\forall a \in D_{Tv_s}(A)\right)\left(Tv^a_s[X \setminus A]^D \cup Tv^a_s[X \setminus A]^I = \varnothing\right.$$
$$\Rightarrow \left(\exists Tv_m \subseteq Tv^a_s[X \setminus A]^M\right)\left(\text{co-nonempty}(Tv_m)\right)\Big)$$
$$\wedge\ Tv_s = \bigcup_{\substack{Tv_m \subseteq Tv^a_s[X \setminus A]^M \\ \text{co-nonempty}(Tv_m) \\ a \in D_{Tv_s}(A)}} Tv_m \cup Tv^a_s[X \setminus A]^D \cup Tv^a_s[X \setminus A]^I\Big).$$

For the proof see the Appendix.

EXAMPLE (cont'd). The set $\{t_3, t_4\}$ is the only co-nonempty set in Tv_3.

C.3. The co-instance set

Suppose that all valuations map Pv-values $t_i.A$ and $t_j.A$ to at most one value. There are two distinct cases as follows:

1) $t_i.A.v \cap t_j.A.v = \varnothing$. Since there exists no relationship between t_i and t_j, it must be that $|(t_i.A.\rho \cup t_j.A.\rho) \cap D(A)| = 1$.
2) $t_i.A.v \cap t_j.A.v \neq \varnothing$. Let $x \in t_i.A.v \cap t_j.A.v$. Since any valuation δ maps x to exactly one value, it must be either $\delta(t_i.A) = \delta(t_j.A)$ or at most one of them is defined.

Thus, the following lemma holds.

LEMMA 4.5. *Let Tv be a nonempty set of Pv-tuples on X. The condition of a co-instance set Tv_s is:*

$$\text{co-instance}(Tv) = (\forall A \in X)\Big(\Big|\bigcup_{t \in Tv} t.A.\rho \cap D(A)\Big| = 1 \vee \left(\exists x \in V(A)\right)\left(Tv = T_{Tv}(A, x)\right)\Big).$$

EXAMPLE (cont'd). The sets $\{t_1, t_3, t_4, t_5\}$ and $\{t_2, t_6\}$ are maximal co-instance sets in Tv_3.

REMARK. Note that $\neg$co–exist($\{t_i, t_j\}$) $\Rightarrow$ co–instance($\{t_i, t_j\}$), for t_i, t_j in Tv. Also note that *co–indefinite($Tv[A]^M$) $\Rightarrow$ co–instance($Tv[A]^M$).*

D. The Formalization of Interpretation

D.1. rep(Tv)

Based on the properties of the Pv-tuple sets discussed above, rep(Tv) can be formalized in a semantically meaningful way.

THEOREM 4.1. *Let Tv^N denote the union of all sets in Tv^M satisfying the condition in Lemma 4.4.*

$$\text{rep}(Tv) = \left\{\text{rep}(Tv^D) \cup \text{rep}(Tv^I \cup Tv^N \cup Tv_{ex}) \,\middle|\, (\exists Tv_{ex} \subseteq Tv^M)(\text{max-co-exist}_{Tv^M}(Tv_{ex}))\right\}$$

$\min-co-exist_{Tv^M}(Tv_{ex})$ is true iff Tv_{ex} is a maximal co-exist set in Tv^M.

For the proof, see the Appendix.

According to Theorem 4.1, to determine whether a relation R is in rep(Tv) is equivalent to determining whether there exists a maximal co-exist set $Tv_{ex} \subseteq Tv^M$ such that $R = \delta(Tv^D) \cup \delta(Tv^I \cup Tv^N \cup Tv_{ex})$. Similarly, to determine whether $R_\varnothing \notin$ rep(Tv) is equivalent to determining whether there exists any definite, indefinite, or co-nonempty set of Pv-tuples in Tv.

D.2. The bound of interpretation

As noted by Lipski [28], the external interpretation has two bounds:

1) A *lower bound* $\|Q\|_l$: the set of (extended) tuples for which we can conclude that they *surely* satisfy Q and
2) An *upper bound* $\|Q\|_h$: the set of (extended) tuples for which we cannot rule out that they *possibly* satisfy Q.

It has been noted that

$$\|Q_1 \vee Q_2\|_l \neq \|Q_1\|_l \cup \|Q_2\|_l \text{ and}$$

$$\|Q_1 \wedge Q_2\|_h \neq \|Q_1\|_h \cap \|Q_2\|_h.$$

In [8], we showed that

$$\|Q_1 \vee Q_2\| = \|Q_1\| \cup \|Q_2\| \text{ and}$$

$$\|Q_1 \wedge Q_2\| = \|Q_1\| \cap \|Q_2\|$$

where $\|Q\|$ corresponds to the query result represented by Pv-table.

If a query Q is formulated on a disjunctive database, we can conclude from the information content that not only definite tuples but also disjunctions of tuples satisfy Q. Moreover, we can also conclude that some conjunctions of tuples cannot satisfy Q simultaneously.

EXAMPLE. Consider Pv-table Tv_4 as the result of query Q shown in Table VII. The lower bound of Q is an empty set while the upper bound is the whole set of tuples. However, from the relationships in Tv_4, we can conclude that either t_1 or t_2 satisfies Q while t_3 and t_4 cannot satisfy Q simultaneously.

TABLE VII
AN ILLUSTRATION OF INTERPRETATION BOUND

Tv_4:	A_1	A_2
t_1:	$(x, a\bar{b})$	$(\varnothing, c)$
t_2	$(x, \bar{a}b)$	$(\varnothing, d)$
t_3	$(wy, a\psi)$	$(\varnothing, e)$
t_4	$(wz, b\psi)$	$(\varnothing, f)$

Alternatively, $\|Q\|_l$ can be defined as the set of relational tuples for which we can conclude that they exist and satisfy Q, and $\|Q\|_h$ as the set of relational tuples for which we cannot rule out the possibility that they exist and satisfy Q. A tighter bounds of interpretation can be defined as:

1) $\|Q\|_{h*}$: the set of nonempty relations for which we cannot rule out the possibility that one of them exists and satisfies Q; and
2) $\|Q\|_{l*}$: the set of *minimal* relations for which we can conclude that exactly one of them exists and satisfies Q

where a minimal relation is a possible relation which does not contain another possible relation.

THEOREM 4.2. *Let Tv represent the result of query Q. We have*

1) $\|Q\|_l = \bigcap_{R\in \text{rep}(Tv)} R = \text{rep}(Tv^D)$;
2) $\|Q\|_{l*} = \{R \mid (R \in \text{rep}(Tv)) \wedge (\nexists R_s \in \text{rep}(Tv))(R_s \subset R)\} \subseteq \text{rep}(Tv^D \cup Tv^I \cup Tv^N)$;
3) $\|Q\|_{h*} = \text{rep}(Tv) \backslash \{R_\varnothing\}$; *and*
4) $\|Q\|_h = \bigcup_{R\in \text{rep}(Tv)} R$.

For the proof, see the Appendix.

REMARK. Let $R_l = \|Q\|_l$ and $R_h = \|Q\|_h$. It is clear that $R_l \subseteq R_{l*}$ for each $R_{l*} \in \|Q\|_{l*}$; and $R_{h*} \subseteq R_h$ for each $R_{h*} \in \|Q\|_{h*}$. Moreover, for each $R_{l*} \in \|Q\|_{l*}$, there exists $R_{h*} \in \|Q\|_{h*}$ such that $R_{l*} \subseteq R_{h*}$.

D.3. The bound of cardinalities of relations in rep(Tv)

Users may be interested in the bound of cardinalities (i.e., the number of tuples) of relations in rep(Tv). The aggregate operation, *count*, is used for this purpose. The bound can be formulated as the following theorem.

THEOREM 4.3. *Let Tv be a Pv-table. Assume that $t_i \neq t_j \Rightarrow \delta(t_i) \neq \delta(t_j)$ for any valuation δ and t_i, t_j in Tv. The cardinalities of relations in* rep(Tv) *range from $|Tv^D| + |Tv^I| + |Tv_{min}|$ to $|Tv^D| + |Tv^I| + |Tv_{max}|$ where Tv_{min} is a minimum subset of Tv^M such that*

$$(\forall Tv_s \subseteq Tv^M)(min-co-nonempty_{Tv^M}(Tv_s) \Rightarrow Tv_{min} \cap Tv_s \neq \varnothing)$$

and Tv_{max} is a maximum subset of Tv^M such that

$$(\forall Tv_s \subseteq Tv^M)(max-co-instance_{Tv^M}(Tv_s) \Rightarrow |Tv_{max} \cap Tv_s| = 1).$$

$min-co-nonempty_{Tv^M}(Tv_s)$ is true iff Tv_s is a minimal co-nonempty set in Tv^M $max-co-instance_{Tv^M}(Tv_s)$ is true iff Tv_s is a maximal co-instance set in Tv^M.

For the proof, see the Appendix.

EXAMPLE. Consider Tv_3 shown in Table V. There are one co-nonempty set, i.e., $\{t_3, t_4\}$ and two co-instance sets, i.e., $\{t_1, t_3, t_4, t_5\}$ and $\{t_2, t_6\}$. Neither definite nor indefinite Pv-tuples is in Tv_3. Hence, the cardinalities of relations in rep(Tv^3) range from 1 to 2.

V. CONCISE REPRESENTATIONS

In this section, we show how to reduce a Pv-table into a more concise form as the query result. By examining the relationships among Pv-tuples, redundant Pv-tuples can be identified easily. In addition, it can be determined whether two Pv-tuples represent the same data and uniteable. Therefore, a Pv-table can be reduced by removing redundant Pv-tuples and uniting Pv-tuples. The same information content is preserved after the reduction. Moreover, the reduction is commutative with relational operations.

Furthermore, it is possible to further condense a Pv-table while preserving the same information content. For example, since rep({<(*uv*, *aψ*)>, <(*xw*, *aψ*)>, <(*xz*, *bψ*)>}) = rep({<(*uv*, *aψ*)>, <(*xz*, *bψ*)>}), <(*xw*, *aψ*)> can be eliminated. However, as the information about some variables (e.g., *w*) will be pruned off, this condensation is not commutative with relational operations. It is only suitable to condense Pv-tables as the results of queries when there are no subsequent operations on these results.

EXAMPLE. Consider Pv-table Tv_3 shown in Table V and Pv-tables Tv_3° and $Tv_3^\bullet$ shown in Table VIII. It can be verified that rep(Tv_3) = rep(Tv_3°) = rep($Tv_3^\bullet$). Tv_3° is the reduction of Tv while $Tv_3^\bullet$ is the condensation of Tv_3°. The difference between Tv_3° and $Tv_3^\bullet$ is that the information about *w* and *z* are pruned off after the condensation.

TABLE VIII
AN ILLUSTRATION OF REDUCTION AND CONDENSATION

Tv_3°:

A
(*wy*, *cψ*)
(*x*, *ab*)
(*zy*, *dψ*)

$Tv_3^\bullet$:

A
(*x*, *ab*)
(*y*, *cdψ*)

A. Reduction

A query on Pv-tables may produce redundancies. The major sources of redundancies come from projection and union operations. If Pv-tuples t_i and t_j represent the same data in a Pv-table and the set of possible tuples represented by t_i is a subset of t_j's, t_i is redundant and can be removed. Also, if there exists a value equivalent Pv-tuple in the same Pv-table for each possible tuple that t_i represents, t_i is redundant and can be removed. Besides, as Pv-tuples may be related with an OR-ship, it is possible to *unite* these Pv-tuples into one. The reduction process is to eliminate redundant Pv-tuples and to unite Pv-tuples.

DEFINITION 5.1. Let t_i and t_j be Pv-tuples in a Pv-table Tv on X. t_i is said to be *redundant*

$$\left(\exists t_j \in Tv\right)\Big(\big(\forall A \in X\big)\big(t_i.A.v \supseteq t_j.A.v \wedge t_i.A.\rho \cap D(A) \subseteq t_j.A.\rho \cap D(A)\big)\Big) \vee \left(\forall t_r \in \text{rep}(t_i)\right)\Big(\left(\exists t_j \in Tv\right)\left(t_r \simeq t_j\right)\Big).$$

EXAMPLE. In the same Pv-table, $<(x, a\bar{b})>$ is redundant if <(*x*, *ab*)> also exists. Further, <(*ux*, *aψ*)> is redundant if either <(*u*, *ac*)> or <(*x*, *ab*)> also exists.

DEFINITION 5.2. *Let t_i, t_j be Pv-tuples in Tv. t_i and t_j are* unite-able *if*

$$\text{OR-ship}_{Tv}(t_i, t_j) \wedge (\exists A_k \in X)(\forall A \in X \backslash A_k)(t_i.A.\rho = t_j.A.\rho).$$

Let t_u be the united tuple of t_i and t_j, which can defined as

$$\left(\forall A \in X\right)\Big(\left(t_u.A.v = t_i.A.v\right) \wedge \Big(t_u.A.\rho = \left(t_i.A.\rho \cup t_j.A.\rho\right) \cap D(A) \cup \left(t_i.A.\rho \cap t_j.A.\rho\right) \cap \left(\overline{D}(A) \cup \{\psi\}\right)\Big)\Big).$$

THEOREM 5.1. rep(Tv) = rep(Tv°) *where Tv° denote the reduction of a Pv-table Tv.*

For the proof, see the Appendix.

EXAMPLE. Consider Pv-table Tv_3 shown in Table V. After the reduction, Pv-tuples (*ux*, *aψ*) and (*xz*, *bψ*) are eliminated and $(x, a\bar{b})$ and $(x, \bar{a}b)$ are united. The result of reduction Tv_3° is shown in Table VIII.

B. Further Condensation of Pv-table

There are two cases that a reduced Pv-table Tv° can be condensed to a more concise Pv-table $Tv^\bullet$ such that rep($Tv^\bullet$) = rep(Tv°) and $|Tv^\bullet| < |Tv^\circ|$. The first case to condense Tv° is to eliminate superfluous Pv-tuples.

DEFINITION 5.3. *Let Tv° be the reduced Pv-table of Tv and $t \in Tv^\circ$. We say t is* superfluous *if* rep(Tv°) = rep($Tv^\circ \backslash t$).

EXAMPLE. Since rep({<(*x*, *ab*)>, <(*y*, *ab*)>, <(*z*, *ab*)>}) = rep({<(*x*, *ab*)>, <(*y*, *ab*)>}), <(*z*, *ab*)> can be eliminated.

For a set of Pv-tuples which are not related with any joinship, as the above example, an efficient algorithm has been proposed to identify and eliminate the superfluous tuples [41].

If Pv-tuples are related with joinship, some superfluous Pv-tuples can be identified as the following lemma states.

LEMMA 5.1. *Let Tv° be the reduced Pv-table of Tv and $t_i \in Tv^\circ$. t_i is superfluous if there exists $t_j \in Tv^\circ$ with the condition:*

$$co-instance\left(\left\{t_i, t_j\right\}\right) \wedge \left(\forall A \in X\right)\left(t_i.A.v \cap t_j.A.v \neq \varnothing \Rightarrow t_i.A.\rho \cap D(A) \subseteq t_j.A.\rho \cap D(A)\right) \wedge \left(\forall t_k \in Tv^\circ \backslash \left\{t_i, t_j\right\}\right)\left(\forall A \in X\right)\left(t_i.A.v \cap t_k.A.v \supseteq t_j.A.v \cap t_k.A.v\right).$$

For the proof see the Appendix.

The other case to condense Tv° is to merge several Pv-tuples into one.

DEFINITION 5.4. *Let Tv° be the reduced Pv-table of Tv. Two Pv-tuples t_i and t_j in Tv° are* mergeable *if* rep($Tv^\circ \backslash \{t_i, t_j\} \cup \{t_m\}$) = rep($Tv^\circ$) *where t_m, the mergence of t_i and t_j, is defined as:*

$$\left(\forall A \in X\right)\Big(\left(t_i.A.v \cap t_j.A.v = \varnothing \Rightarrow t_k.A.v = t_i.A.v \wedge t_k.A.\rho = t_i.A.\rho\right) \wedge \Big(t_i.A.v \cap t_j.A.v \neq \varnothing \Rightarrow t_k.A.v = t_i.A.v \cap t_j.A.v \wedge t_k.A.\rho = \left(t_i.A.\rho \cup t_j.A.\rho\right) \cap D(A) \cup \left(t_i.A.\rho \cap t_j.A.\rho\right) \cap \left(\overline{D}(A) \cup \{\psi\}\right)\Big)\Big).$$

The following lemma states that two Pv-tuples t_i and t_j in Tv° are *mergeable* if the conditions hold:

1) $\{t_i, t_j\}$ is co-instance.

TABLE IX
A TRANSFORMATION FROM A PV-TABLE TO A C-TABLE

Tv_5:	A_1	A_2	A_3
$t1$:	$(x_1, a_1\bar{b}_1)$	(x_2, a_2b_2)	(x_3, a_3b_3)
t_2:	(x_1, a_1b_1)	$(x_2, a_2\bar{b}_2)$	(x_3, a_3b_3)
t_3:	(x_1, a_1b_1)	(x_2, a_2b_2)	$(x_3, a_3\bar{b}_3)$
t_4:	$(y_1z_1, c_1\psi)$	$(w_2, c_2\bar{d}_2)$	$(u_3w_3, c_3\psi)$
t_5:	$(u_1v_1, c_1\psi)$	$(w_2, \bar{c}_2d_2)$	$(v_3w_3, d_3\psi)$

Tc_5:	A_1	A_2	A_3	con
t_1:	x_1	x_2	x_3	$(x_1 = a_1)\wedge(x_2 = a_2 \vee x_2 = b_2)\wedge(x_3 = a_3 \vee x_3 = b_3)$
t_2:	x_1	x_2	x_3	$(x_1 = a_1 \vee x_1 = b_1)\wedge(x_2 = a_2)\wedge(x_3 = a_3 \vee x_3 = b_3)$
t_3:	x_1	x_2	x_3	$(x_1 = a_1 \vee x_1 = b_1)\wedge(x_2 = a_2 \vee x_2 = b_2)\wedge(x_3 = a_3)$
t_4:	y_1	w_2	w_3	$(y_1 = c_1 \wedge y_1 = z_1)\wedge(w_2 = c_2)\wedge(w_3 = c_3 \wedge w_3 = u_3)$
t_5:	u_1	w_2	w_3	$(u_1 = c_1 \wedge u_1 = v_1)\wedge(w_2 = d_2)\wedge(w_3 = d_3 \wedge w_3 = v_3)$

2) t_i and t_j differ from each other in only one Pv-value.
3) The relationships among Pv-tuples should not be changed after the merging process.

Condition 1) follows directly. We illustrate by the following two examples the reason that conditions 2) and 3) must hold.

EXAMPLE. Consider a Pv-table $\{t_1{:}<(x, a\bar{b}), (w, c\bar{d})>, t_2{:}<(x, \bar{a}b), (w, \bar{c}d)>\}$. Suppose that t_1 and t_2 are mergeable, and the mergence of them is $t_m = <(x, ab), (w, cd)>$. As t_1 and t_2 differ from each other in more than one Pv-value, by choosing δ such that $\delta(x) = a$ and $\delta(w) = d$, a relation $\{<a, d>\}$ which is incorrect will be produced.

EXAMPLE. Further, consider the Pv-table $\{t_1 : <(xy, a\psi)>, t_2 : <(xz, b\psi)>, t_3 : <(yz, c\psi)>\}$. Suppose that t_1 and t_2 are mergeable, and the mergence of them is $t_m = <x, ab\psi)>$. $\{t_m, t_3\}$ is co-exist, however, $\{t_2, t_3\}$ is not co-exist. Consequently, $\{t_m, t_3\}$ will cause the producing of $\{<b>, <c>\}$ which is incorrect.

LEMMA 5.2. *Let $Tv°$ be the reduced Pv-table of Tv and t_i and t_j in $Tv°$. t_i and t_j are* mergeable *if the following conditions hold:*

$$co-instance(\{t_i, t_j\})$$
$$\wedge (\exists A_1 \in X)(t_i.A_1.\rho \cap D(A_1) \neq t_j.A_1.\rho \cap D(A_1)$$
$$\wedge (\forall A_2 \in X \setminus A_1)(t_i.A_2.\rho \cap D(A_2) = t_j.A_2.\rho \cap D(A_2)))$$
$$\wedge (\forall t_k \in Tv \setminus \{t_i, t_j\})(t_k.A.v \cap (t_i.A.v \cup t_j.A.v) = t_m.A.v)$$

where t_m is the mergence of t_i and t_j.

For the proof, see the Appendix.

EXAMPLE. Consider Tv_3° shown in Table VIII again. Pv-tuples t_1 and t_3 in Tv_3° are mergeable. The result of condensing Tv_3° is also shown in Table VIII.

C. Transformation

In this subsection, we show how to transform a Pv-table to a C-table [20]. We have shown that the redundant and superfluous Pv-tuples can be identified and removed from a Pv-table. Additionally, the uniteable and mergeable Pv-tuples can be also identified and merged. Yet, a Pv-table may contain Pv-tuples which are co-instance but neither uniteable nor mergeable. Co-instance Pv-tuples can be merged into one after they were transformed to a C-table. A more condensed C-table can therefore be obtained. In considering the transformed C-table as an alternative query result, users should notice that it is not really "equivalent" to the original Pv-table. This is because that indefinite data are no longer distinguishable from maybe data in the C-table.

In the following, we discuss the transformation process. Let λ be a mapping from a Pv-value $t.A$ to its corresponding logical formula in a C-table, defined as

$$\lambda(t.A) = \begin{cases} true & \text{if } t.A.v = \varnothing \\ \left(\bigvee_{a \in t.A.\rho \cap D(A)} (x = a)\right) \wedge \left(\bigwedge_{y \in t.A.v \setminus x} (x = y)\right) & x \in t.A.v \text{ otherwise.} \end{cases}$$

Note that unsatisfiable values are ignored in the mapping. The transformation from a Pv-tuple t to its corresponding C-tuple t_c is defined as

$$t_c.A = \begin{cases} x & \text{if } t.A.v \neq \varnothing \\ a & \text{otherwise} \end{cases} \quad \text{and } t_c.con = \bigwedge_{A \in X} \lambda(t.A)$$

where $x \in t.A.v$ and $\{a\} = t.A.\rho$.

EXAMPLE. Table IX depicts a tuple-by-tuple transformation from a Pv-table Tv_5 to a C-table Tc_5.

In order to obtain a more condensed C-table, the redundant and superfluous Pv-tuples should be identified and removed before the transformation. The transformation is then applied to the maximal co-instance sets corresponding to Tv_{max}. Recall from Section IV.D that Tv_{max} is a maximum subset of Tv^M such that $(\forall\ Tv_s \subseteq Tv)\ (co\text{-}instance(Tv_s) \Rightarrow |Tv_{max} \cap Tv_s| = 1)$. For each maximal co-instance set Tv_s, Pv-tuples in the set are merged into one C-tuple t_c. The merging process is defined as follows.

$$t_c.A = \begin{cases} x & \text{if } x \in \bigcap_{t_i \in Tv_s} t_i.A.v, \\ a & \{a\} = \bigcap_{t_i \in Tv_s} t_i.A.\rho \cap D(A) \text{ otherwise,} \end{cases}$$

TABLE X
A REDUCED C-TABLE

A_1	A_2	A_3	con
x_1	x_2	x_3	$(x_1 = a_1 \vee x_1 = b_1) \wedge (x_2 = a_2 \vee x_2 = b_2) \wedge (x_3 = a_3 \vee x_3 = b_3)$ $\wedge(x_1 = a_1 \vee x_2 = a_2 \vee x_3 = a_3)$
c_1	w_2	w_3	$((y_1 = c_1 \wedge y_1 = z_1) \wedge (w_2 = c_2)(w_3 = c_3 \wedge w_3 = u_3))$ $\vee((u_1 = c_1 \wedge u_1 = v_1) \wedge (w_2 = d_2) \wedge (w_3 = d_3 \wedge w_3 = v_3))$

for each $A \in X$; and

$$t_c.con = \bigvee_{t_i \in Tv_s} t_{ic}.con.$$

Furthermore, if these Pv-tuples are related with an OR-ship, they can be merged in a more readable logical formula.

$$t_c.con = \left(\bigwedge_{A \in X} \left(\bigvee_{a \in D_{Tv_s}(A)} x = a \right) \wedge \left(\bigwedge_{y \in t_i.A.v \setminus x} x = y \right) \right) \wedge \left(\bigvee_{t_i \in Tv_s} \left(\bigwedge_{A \in X} \lambda^*(t_i.A) \right) \right)$$

where

$$\lambda^*(t_i.A) = \begin{cases} \left(\bigvee_{a \in t_i.A.\rho \cap D(A)} x = a \right) & \text{if } |t_i.A.\rho \cap D(A)| \le \left| D_{Tv_s}(A) \right| / 2 \\ \left(\bigwedge_{a \in t_i.A.\rho \cap \overline{D}(A)} x \neq a \right) & \text{otherwise;} \end{cases}$$

and $x \in t_i.A.v$.

It can be shown that the transformation is logically equivalent.

EXAMPLE (cont'd). A reduced C-table which is transformed from Pv-table Tv_5 with a merging process is given in Table X.

VI. CONCLUSION

The relationships among Pv-tuples, namely disjunctive relationship and join relationship, have been explored. Three kinds of Pv-tuple sets were classified according to these relationships. Each set possesses an important property, namely co-exist, co-nonempty or co-instance. Based on these properties, we have shown that the interpretation of Pv-tables can be formalized in a semantically meaningful way. In addition, the lower and upper bounds of interpretation and the range of cardinalities of possible relations can also be characterized.

Moreover, we have shown how to reduce a Pv-table as the query result into a more concise form. The reduction process preserves the same information, and thus is commutative with relational operations. The condensation process can further reduce more Pv-tuples. However, it may lose some relationships among Pv-tuples, and thus is not commutative with relational operations. By examining relationships among Pv-tuples, redundant and superfluous Pv-tuples can be identified and removed. The conditions for determining uniteable and mergeable Pv-tuples, and the process for uniting and merging Pv-tuples were also given.

We also showed how a Pv-table can be transformed to a C-table with a merging process in order to obtain a more condensed query result. The transformation is logically equivalent. However, due to the limitation of C-table, indefinite information will no longer be distinguishable from maybe information.

APPENDIX

PROOF OF LEMMA 4.4. Let Tv_s be the set satisfying the condition. $R_\varnothing \notin$ rep(Tv_s) can be proved by induction with the ground $R_\varnothing \notin$ rep($Tv_s[A]$).

We then show that each minimal co-nonempty set Tv_s satisfies the condition by induction. As the ground, if $Tv_s[A]$ is minimal co-nonempty, it must be co-indefinite on certain variable. Assume that $Tv_s[X \setminus A]$ is co-nonempty iff it satisfies the condition. We want to show that Tv_s is co-nonempty iff it satisfies the condition. Suppose otherwise, Tv_s is minimal co-nonempty but does not satisfy the condition. It implies $Tv_s[A]$ is co-indefinite on x, $Tv_s^a[X \setminus A]^D \cup Tv_s^a[X \setminus A]^I = \varnothing$ and $Tv_s^a[X \setminus A]^M$ is not co-nonempty for some $a \in D_{Tv_s}(A)$. Hence, there exists δ such that $\delta(Tv_s^a[X \setminus A]) = R_\varnothing$. Letting $\delta(t.A) = a$, we have $\delta(Tv_s) = R_\varnothing$, a contradiction. □

PROOF OF THEOREM 4.1. $\subseteq$ part. Let $R = \delta(Tv) \in$ rep(Tv) and Tv_s be the maximal subset of Tv such that $R = \delta(Tv_s)$. It is clear that $(Tv^D \cup Tv^I) \subseteq Tv_s$ and $Tv_s \cap Tv^M$ is co-exist. Thus,

$$Tv_s \subseteq Tv^D \cup Tv^I \cup Tv^N \cup Tv_{ex} \subseteq Tv$$

where $Tv_s \cap Tv^M \subseteq Tv_{ex}$. Consequently,

$$\delta(Tv_s) = \delta(Tv^D \cup Tv^I \cup Tv^N \cup Tv_{ex}) = \delta(Tv) = R.$$

That is, $R \in \text{rep}(Tv^D) \cup \text{rep}(Tv^I \cup Tv^N \cup Tv_{ex})$.

$\supseteq$ part. We claim that there exists δ' such that $\delta(Tv^D \cup Tv^I \cup Tv^N \cup Tv_{ex}) = \delta'(Tv)$. Suppose, otherwise, it must be that $\delta'(Tv \setminus (Tv^D \cup Tv^I \cup Tv^N \cup Tv_{ex}) \neq R_\varnothing$ for any δ' subject to

$$\delta'(Tv^D \cup Tv^I \cup Tv^N \cup Tv_{ex}) = \delta(Tv^D \cup Tv^I \cup Tv^N \cup Tv_{ex}).$$

It implies that there exist $Tv_s \subseteq Tv \setminus (Tv^D \cup Tv^I \cup Tv^N \cup Tv_{ex})$ and $Tv_m \subseteq Tv^N$ such that $co-nonempty(Tv_s \cup Tv_m)$ is true. It causes a contradiction to the definition of Tv^N. □

PROOF OF THEOREM 4.2. (1), (3), and (4) follow directly from their definitions. We only show that $\|Q\|_{l*} \subseteq \text{rep}(Tv^D \cup Tv^I \cup Tv^N)$. That is, for each $R = \delta(Tv) \in \|Q\|_{l*}$, there exists δ' such that $R = \delta'(Tv^D \cup Tv^I \cup Tv^N)$. Suppose otherwise, it must be that $\delta'(Tv \setminus (Tv^D \cup Tv^I \cup Tv^N)) \neq R_\varnothing$ for any δ' subject to

$\delta'(Tv^D \bigcup Tv^I \bigcup Tv^N) = \delta(Tv^D \bigcup Tv^I \bigcup Tv^N)$. It implies that there exist $Tv_s \in Tv \setminus (Tv^D \bigcup TV^I \bigcup Tv^N \bigcup Tv_{ex})$ and $Tv_m \subseteq Tv^N$ such that $co-nonempty(Tv_s \bigcup Tv_m)$ is true. It causes a contradiction to the definition of Tv^N. □

PROOF OF THEOREM 4.3. Min part. We first claim that the minimum number of tuples $\geq |Tv^D| + |Tv^I| + |Tv_{min}|$. Suppose otherwise, there exists $Tv_m \subseteq Tv^M$ such that $|Tv_m| < |Tv_{min}|$ and $\delta(Tv_m) = \delta(Tv^M)$ for some δ. By the definition of Tv_{min}, there must exist $Tv_s \subseteq Tv^M$ such that Tv_s is co-nonempty and $Tv_s \bigcap Tv_m = \varnothing$. Since $\delta(Tv_s \bigcup Tv_m) = \delta(Tv_m)$, we have $\delta(Tv_s) = R_\varnothing$ or $\delta(Tv_s) \subseteq \delta(Tv_m)$, a contradiction. We then claim that there exists δ such that $|\delta(Tv^M)| = |Tv_{min}|$. Suppose otherwise, for any δ, there exists a minimal co-nonempty set Tv_s such that $|\delta(Tv_s)| > 1$. Let $Tv_m = \{t \mid t \in Tv_s \wedge t \notin Tv_{min} \wedge (\exists\delta)(|\delta(Tv_s)| > 1)\}$. We have $\delta(Tv_m) \neq R_\varnothing$ for any δ, i.e., Tv_m is a co-nonempty set. Since $Tv_m > Tv_{min} = \varnothing$, it causes a contradiction to the definition of Tv_{min}. Therefore, there exists $R \in$ rep(Tv) such that $|R| = |Tv^D|+|Tv^I|+|Tv_{min}|$.

Max part. Since any valuation maps a maximal co-instance set to at most one tuple, the maximum number of tuples $\leq |Tv^D| + |Tv^I| + |Tv_{max}|$. We then claim that Tv_{max} is co-exist, which implies $|\delta(Tv_{max})| = |Tv_{max}|$ for some δ. Suppose otherwise, t_i and t_j in Tv_{max} are not co-exist. It implies t_i and t_j are in the same co-instance set, a contradiction to the definition of Tv_{max}. Therefore, there exists $R \in$ rep(Tv) such that $|R| = |Tv^D| + |Tv^I| + |Tv_{max}|$. □

PROOF OF THEOREM 5.1. There are three kinds of reduction processes:

1) t_i is redundant with t_j and has been removed. By Definition 5.1 and Lemma 4.1, we have $\delta(\lambda(t_i))$ is true $\Rightarrow \delta(\lambda(t_j))$ is true for any valuation δ. And, by Definition 4.3, we have $\delta(t_i) = \delta(t_j)$. Hence, we have rep($Tv$) = rep($Tv \setminus t_i$).

2) t_i is redundant with a set of definite Pv-tuples and can be removed. It is clear that rep(Tv) = rep($Tv \setminus t_i$).

3) t_i and t_j are uniteable and have been united as t_u. By Definition 5.2 and Lemma 4.1, we have $\delta(\lambda(t_i)) \vee \delta(\lambda(t_j))$ is true $\Rightarrow \delta(\lambda(t_u))$ is true for any valuation δ. In addition, by Definition 4.3, we have $\delta(t_i) = \delta(t_u)$ if $\delta(t_i)$ is defined. Symmetrically, $\delta(t_j) = \delta(t_u)$ if $\delta(t_j)$ is defined. Hence, we have rep(Tv) = rep($Tv \setminus \{t_i, t_j\} \cup t_u$). □

PROOF OF LEMMA 5.1. To show rep(Tv°) = rep($Tv^\circ \setminus t_i$) it is sufficient to show 1) rep($\{t_i, t_j\}$) = rep($\{t_j\}$) and 2) rep($\{t_i, t_j, t_k\}$) = rep($\{t_j, t_k\}$) for all t_k in $Tv^\circ \setminus \{t_i, t_j\}$. According to Lemma 4.1 and Definition 4.3, the first two conditions ensure that if $\delta(t_i)$ is defined, there exists δ' such that $\delta(t_i) = \delta'(t_j)$. Hence, 1) holds. To prove 2), we note that the last two conditions ensure that $co\text{-}exist(\{t_i, t_k\}) \Rightarrow co\text{-}exist(\{t_j, t_k\})$. That is, if there exists a valuation δ such that both $\delta(t_i)$ and $\delta(t_k)$ are defined, there must exist another valuation δ' such that both $\delta'(t_j)$ and $\delta'(t_k)$ are defined. Let δ' be subject to $\delta'(t_k) = \delta(t_k)$ and $\delta'(x) = \delta(x)$ for $x \in t_i.A.v \bigcap t_j.A.v$. We have $\delta(t_i) = \delta'(t_j)$. Hence, 2) holds. □

PROOF OF LEMMA 5.2. To prove

$$rep(Tv^\circ) = rep(Tv^\circ \setminus \{t_i, t_j\} \bigcup t_m)$$

it is sufficient to show 1) rep($\{t_i, t_j\}$) = rep($\{t_m\}$) and 2) rep($\{t_i, t_j, t_k\}$) = rep($\{t_m, t_k\}$) for all t_k in $Tv^\circ \setminus \{t_i, t_j\}$. According to Lemma 4.1 and Definition 4.3, if either $\delta(t_i)$ or $\delta(t_j)$ is defined, the first condition ensures that there exists δ' such that either $\delta(t_i) = \delta'(t_m)$ or $\delta(t_j) = \delta'(t_m)$. Conversely, if $\delta'(t_m)$ is defined, the first two conditions ensure that there exists δ subject to $\delta(x) = \delta'(x)$, $x \in t_m.A.v$, $A \in X$. $\delta(t_i) = \delta'(t_m)$ if $\delta'(x) \in t_i.A_1.\rho \bigcap D(A)$, $\delta(t_j) = \delta'(t_m)$ otherwise. Hence, 1) holds. To prove 2), we note that the last condition ensures that $co\text{-}exist(\{t_m, t_k\}) \Leftrightarrow co\text{-}exist(\{t_i, t_k\}) \wedge co\text{-}exist(\{t_j, t_k\}))$. Moreover, there exists a valuation δ such that both $\delta(t_i)$ and $\delta(t_k)$ are defined, for a similar reason in proving 1), iff there exists another valuation δ' such that both $\delta'(t_m)$ and $\delta'(t_k)$ are defined, and $\delta(t_i) = \delta'(t_m)$ and $\delta(t_k) = \delta'(t_k)$. Symmetrically, so do t_j and t_k. Hence, (2) holds. □

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for their useful suggestions and comments.

This research was partially supported by the Republic of China National Science Council under Contract No. NSC 84-2213-E-007-007.

REFERENCES

[1] S. Abiteboul and G. Grahne, "Update semantics for incomplete databases," *Proc. VLDB Conf.*, pp. 1-12, 1985.

[2] S. Abiteboul, P. Kanellakis, and G. Grahne, "On the representation and querying of sets of possible worlds," *Proc. ACM SIGMOD Conf.*, pp. 34-48, 1987.

[3] D. Barbará, H. Garcia-Molina, and D. Porter, "The management of probabilistic data," *IEEE Trans. Knowledge and Data Engineering*, vol. 4, no. 4, pp. 487-501, 1992.

[4] J. Biskup, "Foundations of Codd's relational maybe operations," *ACM Trans. Database Systems*, vol. 8, no. 4, pp. 608-636, 1983.

[5] B.P. Buckles and F.E. Petry, "A fuzzy representation of data for relational databases," *Fuzzy Sets and Systems*, vol. 7, pp. 213-226, 1982.

[6] R. Cavallo and M. Pittarelli, "The theory of probabilistic databases," *Proc. VLDB Conf.*, pp. 71-81, 1987.

[7] J.S. Chiu and A.L.P. Chen, "A sound and complete extended relational algebra for exclusive disjunctive data," *The Poster Paper Collection, VLDB Conf.*, pp. 12-22, 1994.

[8] J.S. Chiu and A.L.P. Chen, "Pv-table—a representation system for exclusive disjunctive data," Technical Report, Dept. of Computer Science, Nat'l Tsing Hua Univ., 1994.

[9] E. Codd, "Understanding relations," Installment no. 7, *ACM SIGMOD Record FDT Bulletin*, vol. 7, no. 3-4, pp. 23-28, 1975.

[10] E. Codd, "Extending the database relational model to capture more meaning," *ACM Trans. Database Systems*, vol. 4, no. 4, pp. 397-434, 1979.

[11] E. Codd, "Missing information (applicable and inapplicable) in relational databases," *ACM SIGMOD Record*, vol. 15, no. 4, pp. 53-78, 1986.

[12] 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, pp. 485-493, 1989.

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

[14] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco: W.H. Freeman and Co., 1979.

[15] G. Grahne, "Dependency satisfaction in databases with incomplete information," *Proc. VLDB Conf.*, pp. 37-45, 1984.

[16] J. Grant, "Partial values in a tabular database model," *Information Processing Letters*, vol. 9, no. 2, pp. 97-99, 1979.

[17] J. Grant, "Incomplete information in a relational database," *Fundamenta Informaticae*, vol. 3, no. 3, pp. 363-378, 1980.

[18] J. Grant and J. Minker, "Answering queries in indefinite databases and the null value problem," *Advances in Computing Research*, P. Kanellakis, ed., *The Theory of Databases*, vol. 3, pp. 247-267. JAI Press, 1986.

[19] T. Imielinski and W. Lipski Jr., "Incomplete information and dependencies in relational databases," *Proc. ACM SIGMOD Conf.*, pp. 178-184, 1983.

[20] T. Imielinski and W. Lipski Jr., "Incomplete information in relational databases," *J. ACM*, vol. 31, no. 4, pp. 761-791, 1984.

[21] T. Imielinski, S. Naqvi, and K. Vadaparty, "Incomplete objects—A data model for design and planning applications," *Proc. ACM SIGMOD Conf.*, pp. 288-297, 1991.

[22] T. Imielinski and K. Vadaparty, "Complexity of query processing in databases with OR-objects," *Proc. ACM PODS Conf.*, pp. 51-65, 1989.

[23] M. Koubarakis, "Database models for infinite and indefinite temporal information," *Information Systems*, vol. 19, no. 2, pp. 141-173, 1994.

[24] M. Levene and G. Loizou, "Semantics for null extended nested relations," *ACM Trans. Database Systems*, vol. 18, no. 3, pp. 414-459, 1993.

[25] H.J. Levesque, "The interaction with incomplete knowledge bases: A formal treatment," *Proc. Int'l Joint Conf. Artificial Intelligence*, pp. 240-245, 1981.

[26] L. Libkin and L. Wong, "Semantic representations and query languages for or-sets," *Proc. ACM PODS Conf.*, pp. 37-48, 1993.

[27] Y.E. Lien, "Multivalued dependencies with null values in relational data bases," *Proc. VLDB Conf.*, pp. 61-66, 1979.

[28] W. Lipski, "On semantic issue connected with incomplete information systems," *ACM Trans. Database Systems*, vol. 4, no. 3, pp. 262-296, 1979.

[29] K.C. Liu and R. Sunderraman, "Indefinite and maybe information in relational databases," *ACM Trans. Database Systems*, vol. 15, no. 1, pp. 1-39, 1990.

[30] 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, pp. 65-77, 1991.

[31] D. Maier, *The Theory of Relational Databases*. Rockville, Md.: Computer Science Press, 1983.

[32] J. Minker, "On indefinite databases and the closed world assumption," *Proc. Automated Deduction Conf.*, pp. 292-308, 1982.

[33] J.M. Morrissey, "Imprecise information and uncertainty in information systems," *ACM Trans. Information Systems*, vol. 8, no. 2, pp. 159-180, 1990.

[34] A. Ola, "Relational databases with exclusive disjunctions," *Proc. IEEE Data Eng. Conf.*, pp. 328-336, 1992.

[35] A. Ola and G. Ozsoyoglu, "Incomplete relational database models based on intervals," *IEEE Trans. Knowledge and Data Engineering*, vol. 5, no. 2, pp. 293-308, 1993.

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

[37] R. Reiter, "On closed world databases," *Logic and Databases*, H. Gallaire and J. Minker, eds. , pp. 55-76. New York: Plenum Press, 1978.

[38] R. Reiter, "Towards a logical reconstruction of relational database theory," *On Conceptual Modelling*, M.L. Brodie, J. Mylopoulos, and J.W. Schmidt, eds., pp. 191-233. New York: Springer-Verlag, 1984.

[39] R. Reiter, "A sound and sometimes complete query evaluation algorithm for relational databases with null values," *J. ACM*, vol. 33, no. 2, pp. 349-370, 1986.

[40] 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 Int'l J.*, pp. 281-302, 1993.

[41] 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 VLDB J.*, vol. 2, no. 4, pp. 489-512, 1993.

[42] F.S.C. Tseng, A.L.P. Chen and W.P. Yang, "Refining imprecise data by integrity constraints," *Data and Knowledge Eng. J.*, vol. 11, no. 3, pp. 299-316, 1993.

[43] Y. Vassiliou, "Null values in database management: A denotational semantics approach," *Proc. ACM SIGMOD Conf.*, pp. 162-169, 1979.

[44] Y. Vassiliou, "Functional dependencies and incomplete information," *Proc. VLDB Conf.*, pp. 260-269, 1980.

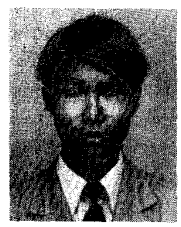
[45] M. Winslett, "A model-based approach to updating databases with incomplete information," *ACM Trans. Database Systems*, vol. 13, no. 2, pp. 167-196, 1988.

[46] L.C. Yuan and D.A. Chiang, "A sound and complete query evaluation algorithm for relational databases with null values," *Proc. ACM SIGMOD Conf.*, pp. 74-81, 1988.

[47] L.A. Zadeh, "Fuzzy sets as a basis for a theory of possibility," *Fuzzy Sets and Systems*, vol. 1, no. 1, pp. 3-28, 1978.

[48] C. Zaniolo, "Database systems with null values," *J. Computer and System Sciences*, vol. 28, pp. 142-166, 1984.

[49] M. Zemankova, "Implementing imprecision in information systems," *Information Science*, vol. 37, pp. 107-141, 1985.



Jui-Shang Chiu received the MS degree in computer science from National Tsing Hua University, Hsinchu, Taiwan, Republic of China, in 1989. He is currently a candidate for the PhD degree in computer science at National Tsing Hua University. His research interests include databases, artificial intelligence, and algorithms.

Arbee L.P. Chen received the BS degree in computer science from National Chiao Tung University, Taiwan, Republic of China, in 1977,and the PhD degree in computer science from the University of Southern Calfornia, Los Angeles, in 1984.

Dr. Chen joined National Tsing Hua University, Taiwan, as a visiting specialist in August 1990, and became a professor in the Department of Computer Science in 1991. He was a member of the technical staff at Bell Communications Research, New Jersey, from 1987 to 1990; an adjunct associate professor in the Department of Electrical Engineering and Computer Science, Polytechnic University, Brooklyn, New York; and a research scientist at Unisys, Santa Monica, California, from 1985 to 1986. He is currently director of the Computer and Communications Center of National Tsing Hua University, and advisor to the Industrial Technology Research Institute and the Institute for Information Industry. His research interests include heterogeneous database systems, incomplete information, object-oriented databases, and multimedia databases.

Dr. Chen is a member of the ACM and the IEEE Computer Society. He served as a program co-chair for the IEEE Data Engineering Conference, and as a program committee member for the ACM SIGMOD Conference, the IEEE Data Engineering Conference, and the VLDB, CIKM, DASFAA, ICPADS, COMAD, and DKSME conferences. He is listed in Marquis' *Who's Who in the World*.