Incomplete Information in Relational Databases

TOMASZ IMIELIŃSKI AND WITOLD LIPSKI, JR.

Polish Academy of Sciences, Warsaw, Poland

Abstract. This paper concerns the semantics of Codd's relational model of data. Formulated are precise conditions that should be satisfied in a semantically meaningful extension of the usual relational operators, such as projection, selection, union, and join, from operators on relations to operators on tables with "null values" of various kinds allowed. These conditions require that the system be safe in the sense that no incorrect conclusion is derivable by using a specified subset Ω of the relational operators; and that it be complete in the sense that all valid conclusions expressible by relational expressions using operators in Ω are in fact derivable in this system. Two such systems of practical interest are shown. The first, based on the usual Codd's null values, supports projection and selection. The second, based on many different ("marked") null values or variables allowed to appear in a table, is shown to correctly support projection, positive selection (with no negation occurring in the selection condition), union, and renaming of attributes, which allows for processing arbitrary conjunctive queries. A very desirable property enjoyed by this system is that all relational operators on tables are performed in exactly the same way as in the case of the usual relations. A third system, mainly of theoretical interest, supporting projection, selection, union, join, and renaming, is also discussed. Under a so-called closed world assumption, it can also handle the operator of difference. It is based on a device called a conditional table and is crucial to the proof of the correctness of the second system. All systems considered allow for relational expressions containing arbitrarily many different relation symbols, and no form of the universal relation assumption is required.

Categories and Subject Descriptors: H.2.3 **[Database Management]**: Languages—*query languages*; H.2.4 **[Database Management]**: Systems—*query processing*

General Terms: Theory

Additional Key Words and Phrases: Relational database, incomplete information, null values, marked nulls, relational algebra, query language semantics, query processing

1. *Introduction*

Attempts to represent incomplete information in the relational model of data have been made since the very beginning of the relational database theory [3, 7, 8, 10, 12, 13, 18-21, 27, 28, 30, 31, 33, 34]. (See also the extensive bibliography in [22].) The main device in this context has been the *null value*, a special symbol @

This work was supported in part by the Polish Academy of Sciences under Contract MR.I.3. The second author was also aided through a visiting appointment by Istituto di Analisi dei Sistemi ed Informatica, Consiglio Nazionale delle Richerche, Roma, Italy.

Some of the results of this paper were presented at the Seventh International Conference on Very Large Data Bases, Cannes, France, September 1981.

Authors' present addresses: T. Imieliński, Rutgers University, Department of Computer Science, Hill Center for the Mathematical Sciences, Busch Campus, New Brunswick, NJ, 08903; W. Lipski, Jr., Laboratoire de Recherche en Informatique, Université de Paris-Sud, Centre d'Orsay, Bât. 490, 91405 Orsay Cedex, France.

allowed as an entry in a table, with its intended meaning being *value at present unknown* (*but the attribute applicable*).

Representing incomplete information in a database immediately raises the much more difficult and important problem of processing this information so that the user can obtain—perhaps partial, but correct—responses to his or her queries on the basis of the incomplete information. In the context of the relational model, this comes down to defining the relational operators on tables with null values in a semantically correct way. The intuitive condition that should be satisfied is the following. If a null value has a specified semantic interpretation, that is, if we assume that a specified relation exists between a table with nulls and the real world, then this relation should be similar for the tables that are arguments of a relational operator and for the table obtained as the result.

Solutions to this problem proposed so far have been in many respects partial and unsatisfactory. Codd's approach [7, 8] is based on a three-valued logic and a so-called null substitution principle, which has been criticized on semantic grounds by Grant [10] and Lipski [20]. Lipski [19–21] and Vassiliou [30] consider, essentially, only the relational operator of selection, where the selection condition can be an arbitrary Boolean combination of atomic conditions. Proposals involving a richer subset of the relational operators—and it is the join that seems to be the main source of difficulty—usually do not give a clear and precise explanation of the sense in which the definitions of relational operators on tables with nulls are semantically correct [7, 8, 18]. An exception is the work of Biskup [3]. However, his definition of, say, a join $T \bowtie U$ of two tables with nulls works correctly when T and U are "independent" but does not necessarily provide a method for correctly evaluating an expression $f(T) \bowtie g(T)$ where f and g are some relational expressions. This is because in the latter case there may be "additional information" implied by the fact that both arguments of the join come from the same table T, and his definition of join is not able to take this into account (this is discussed in more detail at the end of Section 4.) In fact, queries in a relational database are arbitrary relational expressions, so that we should be able to handle correctly relational expressions rather than just single relational operators.

In this paper we formulate the precise conditions that we believe should be satisfied in any semantically meaningful extension of the usual relational operators, such as projection, selection, union, and join, from operators on relations to operators on "incompletely specified relations"—typically, tables with "null values" of different kinds allowed as entries. These conditions are embodied into the definition of a representation system (see Section 3).

Roughly speaking, these conditions require that our system be safe in the sense that no incorrect conclusion is derivable by using a specified subset Ω of the relational operators, and that it be complete in the sense that all valid conclusions expressible by relational operators using operators in Ω are in fact derivable. The intuition behind the notion of a representation system can also be explained in the following way. By performing relational operators over tables with nulls, we may introduce some corruption of information. However, if the query language that the user has at his or her disposal is weak enough, that is, if only a specified subset Ω of the relational operators is available, then this corruption is not visible by the user.

Two representation systems of practical interest are shown. The first, described in Section 4, is based on tables with the usual Codd null values @ (we call them Codd tables). The following is an example of a Codd table.

SUPPLIER	LOCATION	PRODUCT	QUANTITY
Smith	London	Nails	@
Brown	@	Bolts	@
Jones	@	Nuts	40,000

This table represents the information that Smith, located in London, supplies (an unknown quantity of) nails; Brown supplies bolts; and Jones supplies 40,000 nuts. The system based on Codd tables supports projection and selection, and it is shown that no representation system based on Codd tables can support projection and join at a time.

Section 5 describes some algorithms for correctly evaluating a selection over a Codd table, including the case in which attribute domains are linearly ordered, and the selection condition is an arbitrary Boolean combination of elementary inequalities between attributes or between attributes and values.

The second representation system, presented in Section 6, supports projection, positive selection (with no negation occurring in the selection condition), union, join, and renaming of attributes—this allows for processing arbitrary conjunctive queries [4], as shown in Section 8. It is based on tables allowing many different ("marked") null values, or variables (such tables are called *V*-tables). The following is an example of a *V*-table:

COURSE	TEACHER	WEEKDAY
Databases	x	Monday
Programming	y	Tuesday
Databases	x	Thursday
FORTRAN	Smith	z

This table contains three different variables, x, y, z. Notice that it represents the information that the teacher of the course on databases, though unknown, is the same on Monday and Thursday, a fact not representable by means of a Codd table.

A very desirable property enjoyed by our representation system based on *V*-tables is that all relational operators on *V*-tables are performed in exactly the same way as in the case of the usual relations, treating variables as if they were regular values in appropriate attribute domains (this may be referred to as the *naive evaluation*). It may also be noted that *V*-tables appear in a natural way in the context of updating relational views. Assume, for instance, that a database is given by

SL :

SUPPLIER	LOCATION
Smith	London

SP :

SUPPLIER	PRODUCT
Smith	Nails

and suppose that the relational view

$$\pi_{\text{LOCATION, PRODUCT}}(\text{SL} \bowtie \text{SP})$$

is updated by adding two tuples, $\langle$New York, Bolts$\rangle$ and $\langle$Los Angeles, Nuts$\rangle$. It is then natural to define the effect of this update to be

SL :

SUPPLIER	LOCATION
Smith	London
x	New York
y	Los Angeles

SP :

SUPPLIER	PRODUCT
Smith	Nails
x	Bolts
y	Nuts

A negative result shown in Section 6 is that, rather surprisingly, no representation system based on V-tables can support projection and arbitrary selection (recall that these operators can be supported by Codd tables).

A third representation system, described in Section 7, supports projection, selection, union, join, and renaming and is mainly of a theoretical interest. It is based on an idea of a so-called conditional table and is crucial in the proof of the correctness of the second system. A conditional table is a V-table with an additional column, *con*, containing a condition; for example, the conditional table

SUPPLIER	LOCATION	PRODUCT	con
x	London	Nails	$x =$ Smith
Brown	New York	Nails	$x \neq$ Smith

represents the information that nails are supplied either by Smith in London or by Brown in New York, but not by both at a time.

All representation systems considered in this paper allow for relational expressions containing arbitrarily many different relation symbols, and no form of the universal relation assumption [24] is required.

Most of the results of the paper are developed under a so-called open world assumption (see [26]) that, roughly speaking, means that we are not able to represent negative information. In Section 9 we briefly show how the results can be extended in a straightforward manner to a modified interpretation of tables where negative information is representable. It turns out that under such a modified interpretation the system based on conditional tables can also handle the operator of difference, thus supporting the full strength of the relational algebra.

The last section contains conclusions and briefly describes related work, in particular, the work concerning the problem of handling dependencies in the context of V-tables, and the relation between V-tables and tableaux of Aho, Sagiv, and Ullman [1].

2. *Basic Definitions*

In this section, we give some basic definitions and notation concerning the relational data model ([5]; see also [29]).

Throughout the paper we consider a fixed, finite (unless otherwise stated) set $\mathcal{U}$ of *attributes*. Attributes are usually denoted by A, B, C, and sets of attributes by X, Y, Z, with possible subscripts. A set of attributes, say $\{A, C\}$, is usually written as AC. Associated with every $A \in \mathcal{U}$ is an *attribute domain* $D(A)$. We always assume that $|D(A)| \geq 2$ and we denote $D = \bigcup_{A \in \mathcal{U}} D(A)$. Elements of D are sometimes called *constants*. Elements of $D(A)$, $D(B)$, $D(C)$ are usually denoted by a, b, c, respectively, with possible primes, etc. By a *tuple* on X we mean any mapping t that associates a value $t(A) \in D(A)$ with every $A \in X$. A tuple is usually denoted as a string of values associated with the attributes; for example, ac is a tuple on AC. For a tuple t on X and for any $Y \subseteq X$, we denote by $t[Y]$ the *restriction* of t to Y; for example, if $t = abc$ then $t[AC] = ac$. By a *relation* on X we mean any finite set of tuples on X. If t, r are a tuple and a relation on X, we then write

$$\alpha(t) = \alpha(r) = X,$$

and we call $\alpha(t)$ and $\alpha(r)$ the *type* of t and r, respectively. A *multirelation* of type $\langle X_1, \ldots, X_n \rangle$ is any sequence $\langle r_1, \ldots, r_n \rangle$ in which r_i is a relation on X_i, $1 \le i \le n$. Relations are usually denoted by r, s, with possible subscripts, and multirelations by $\mathbf{r}$, $\mathbf{s}$ (conforming to the general convention that boldface indicates multiobjects). The type of multirelation $\mathbf{r}$ is denoted by $\alpha(\mathbf{r})$. By an empty multirelation we mean any multirelation of the form $\mathbf{0} = \langle \varnothing, \ldots, \varnothing \rangle$; formally, we shall assume that there is a different empty multirelation for each type.

For two multirelations $\mathbf{r} = \langle r_1, \ldots, r_n \rangle$, $\mathbf{s} = \langle s_1, \ldots, s_n \rangle$ of the same type, we write $\mathbf{r} \subseteq \mathbf{s}$ if $r_i \subseteq s_i$, $1 \le i \le n$. If $\mathbf{r} = \langle r_1, \ldots, r_n \rangle$ and $1 \le i \le n$, then we define

$$\mathrm{pr}_i(\mathbf{r}) = r_i.$$

The set of all multirelations of type $\langle X_1, \ldots, X_n \rangle$ is denoted by $\mathscr{R}(X_1, \ldots, X_n)$; in particular, $\mathscr{R}(X)$ is the set of all relations of type X. The set of all multirelations is denoted by $\mathscr{R}$.

The class of all nonempty homogeneous sets of multirelations is denoted by $\mathscr{I}$ (a set $\mathscr{X}$ of multirelations is *homogeneous* if $\alpha(\mathbf{r}) = \alpha(\mathbf{s})$ for all $\mathbf{r}, \mathbf{s} \in \mathscr{X}$). Elements of $\mathscr{I}$ are denoted by $\mathscr{X}$, $\mathscr{Y}$, $\mathscr{Z}$, and the definition of $\alpha(\mathscr{X})$, the type of $\mathscr{X}$, is the natural one.

We consider the usual relational operators:

Projection

$$\pi_Y(r) = \{t[Y] : t \in r\} \qquad (Y \subseteq \alpha(r)).$$

Selection

$$\sigma_E(r) = \{t \in r : E(t) = \mathbf{true}\}.$$

Here E is a *selection condition*; that is, any expression built up from atomic conditions of the form $(A = a)$, $(A = B)$, $A, B \in \mathscr{U}$, $D(A) = D(B)$, $a \in D(A)$ (and truth constants **true**, **false**) by means of the logical connectives $\vee$ (or), $\wedge$ (and), $\neg$ (not). $E(t)$ denotes the truth value obtained by substituting the value $t(A)$ for any occurrence of A in E and evaluating the expression in the natural way. We always assume that $\alpha(r)$ contains all the attributes occurring in E. A selection condition E and the selection operator σ_E is called *positive* if it does not contain the $\neg$ symbol.

We sometimes consider a more general form of selection, where E is *any* mapping from tuples to {**true**, **false**}. (Note: The operation $\sigma_{A=B}$ is also called *restriction*.)

Union

$r \cup s$, that is, the usual set-theoretical union.

We always assume that both arguments of union are of the same type.

Join

$$r \bowtie s = \{t : \alpha(t) = X \cup Y \wedge t[X] \in r \wedge t[Y] \in s\},$$

where

$$X = \alpha(r), \qquad Y = \alpha(s).$$

Difference

$r - s = r \backslash s$, that is, the usual set-theoretical difference.

We always assume that both arguments of difference are of the same type.

In Section 8 we also consider the following additional operation.

Renaming (of an attribute)

$$s_B^A(r) = \{s_B^A(t) : t \in r\}$$

where $A \in \alpha(r)$, $B \in \mathscr{U}\backslash\alpha(r)$, $D(A) = D(B)$, and $s_B^A(t)$ is the tuple of type $(\alpha(r)\backslash\{A\}) \cup \{B\}$ with

$$(s_B^A(t))(C) = \begin{cases} t(C) & \text{if } C \in \alpha(r)\backslash\{A\}, \\ t(A) & \text{if } C = B. \end{cases}$$

Intuitively, $s_B^A(r)$ is the result of renaming column A or r to B.

We show in Section 8 how the results of this paper can be extended to another version of relational algebra, based on Cartesian product (see, e.g., [29]) instead of join.

By a *relation name* we mean a symbol R (the letter S is also used) with possible subscripts, with an associated *type* $\alpha(R) \subseteq \mathscr{U}$.

An *instance* of R is any relation r such that $\alpha(r) = \alpha(R)$. An instance is usually denoted by a lowercase version of the letter denoting the relation name.

For any subset Ω of the relational operators of projection (P), selection (S), positive selection (S$^+$), join (J), difference (D), and renaming (R), a *relational Ω-expression* means any well-formed expression built up from relation names and relational operators in Ω; for example, $\sigma_{A=a}(R_1) \bowtie \pi_{AB}(R_2)$ is a relational PS$^+$J-expression. A *multirelational Ω-expression* is any sequence $\mathbf{f} = \langle f_1, \ldots, f_k \rangle$ of relational Ω-expressions. Relational and multirelational Ω-expressions are often called simply Ω-expressions; they are usually denoted by f, g, with possible subscripts, and by $\mathbf{f}$, $\mathbf{g}$, respectively. They are assumed to be *typed* in the following sense. We assume that associated with every $\mathbf{f}$ is a sequence $\langle R_1, \ldots, R_n \rangle$ (either clear from the context or given explicity by writing $\mathbf{f}\,(R_1, \ldots, R_n))$ containing all relation names occurring in $\mathbf{f}$ (and possibly some other relation names). The *argument type* of $\mathbf{f}$ is then defined as

$$\alpha(\mathbf{f}) = \langle \alpha(R_1), \ldots, \alpha(R_n) \rangle.$$

For any $\mathbf{f} = \langle f_1, \ldots, f_k \rangle$ and any multirelation $\mathbf{r} = \langle r_1, \ldots, r_n \rangle$ of type $\alpha(\mathbf{r}) = \alpha(\mathbf{f})$, we define $\mathbf{f}(\mathbf{r}) = \langle f_1(\mathbf{r}), \ldots, f_k(\mathbf{r}) \rangle$ where $f_i(\mathbf{r})$ denotes the relation obtained by substituting r_j for all occurrences of R_j in f_i, $j = 1, \ldots, n$. We also define $\beta(\mathbf{f}) = \alpha(\mathbf{f}(\mathbf{r}))$ and call it the *result type* of $\mathbf{f}$.

It may be mentioned in this context that the relational algebra can be embedded in a natural way into a simple algebraic system called a *cylindric set algebra*, such that all relations considered are of type $\mathscr{U}$ and all relational operators are total (i.e., there are no restrictions on the arguments). This approach has several advantages (i.e., the "complicated" join operator becomes the usual set-theoretical intersection) but leads to infinite relations and creates some problems concerning the finite representability of these relations. This approach is treated in more detail in another paper by the authors [14].

3. *Representation Systems*

In this section we formulate some general conditions that we require to be satisfied in any semantically meaningful extension of the relational algebra to "tables with nulls."

To this end we introduce the notion of a representation system. By a *representation system* we mean a triple $\langle \mathscr{T}, \mathrm{Rep}, \Omega \rangle$ where $\mathscr{T}$ is the set of *multitables*, Rep is a mapping, Rep: $\mathscr{T} \to \mathscr{I}$ (recall that $\mathscr{I}$ is the set of all nonempty homogeneous sets of multirelations), Ω is a set of relational operators, and a certain natural condition, to be defined at the end of this section, is satisfied. Roughly speaking, this condition asserts that there is a way, consistent with respect to operators in Ω, to define $\mathbf{f}(\mathbf{T})$ for any Ω-expression $\mathbf{f}$ and any multitable $\mathbf{T} \in \mathscr{T}$.

First, however, let us give some intuitive explanations concerning the components of a representation system.

A multitable is usually a "generalized multirelation" with tuples allowed to contain not only values in appropriate attribute domains, but also some special symbols ("null values" of various kinds). Most of the notions and notation concerning relations and multirelations, such as a tuple, $\alpha(\mathbf{r})$, or $t[Y]$, carry over to multitables. The set Rep(T) of multirelations defines the information contained in multitable T; that is, it specifies the set of possibilities represented by T.

Suppose $\mathscr{X} \in \mathscr{I}$ represents our information about a certain unknown $\mathbf{r}^* \in \mathscr{X}$ and let $\mathbf{f}$ be an Ω-expression with $\alpha(\mathbf{f}) = \alpha(\mathscr{X})$. Clearly, our information about $\mathbf{f}(\mathbf{r}^*)$ is then represented by

$$\mathbf{f}(\mathscr{X}) = \{\mathbf{f}(\mathbf{r}) : \mathbf{r} \in \mathscr{X}\}.$$

In this way, any Ω-expression $\mathbf{f}$ can be treated in a natural way as a mapping $\mathbf{f}$: $\mathscr{I} \to \mathscr{I}$ (more exactly, a partial mapping defined only for those $\mathscr{X} \in \mathscr{I}$ with $\alpha(\mathscr{X}) = \alpha(\mathbf{f})$).

Suppose now that we want to give a natural definition of multitable f(T) where $\mathbf{T} \in \mathscr{T}$ and $\mathbf{f}$ is an Ω-expression ($\alpha(\mathbf{T}) = \alpha(\mathbf{f})$, where $\alpha(\mathbf{T})$ can be defined to be $\alpha(\mathrm{Rep}(\mathbf{T}))$). If we think of T as representing in an incomplete way some unknown $\mathbf{r}^* \in \mathrm{Rep}(\mathbf{T})$, then ideally we could expect that there exist the same relation between f(T) and $\mathbf{f}(\mathbf{r}^*)$, so that f(Rep(T)) = Rep(f(T)); that is, the following diagram commutes:

$$\begin{tikzcd} \mathscr{T} \arrow[r, "\mathrm{Rep}"] \arrow[d, dashed, "\mathbf{f}"'] & \mathscr{I} \arrow[d, "\mathbf{f}"] \\ \mathscr{T} \arrow[r, "\mathrm{Rep}"'] & \mathscr{I} \end{tikzcd} \tag{3.1}$$

(for multitables of appropriate type). Unfortunately, this approach is usually not feasible, since in most practical situations the structure of the set f(Rep(T)) is not "regular" enough to be representable by any $\mathbf{U} \in \mathscr{T}$ (see, however, an exception at the end of Section 9). In other words, there is no $\mathbf{U} \in \mathscr{T}$ such that Rep(U) = f(Rep(T)).

Clearly, if we are to define f(T) in a semantically meaningful way, then we should require that Rep(f(T)) approximate the information given by f(Rep(T)) in some natural sense (and we should precisely state in which sense). Before we describe what we consider to be a natural notion of "equivalence" between Rep(f(T)) and f(Rep(T)), we give some examples illustrating different kinds of equivalences between sets of multirelations.

Example 3.1. Let $\mathscr{X}$ consists of all those relations that contain at least one of the following two relations:

$$\boxed{\begin{matrix} a & b \\ a' & b' \end{matrix}}\,, \qquad \boxed{\begin{matrix} a & b \\ a' & b'' \end{matrix}}\,.$$

(Note: Here, and in all other examples in this paper, different symbols denoting values in attribute domains always stand for different values.)

Let $\mathscr{Y}$ be the set of all relations containing tuple ab (clearly $\mathscr{X} \subsetneq \mathscr{Y}$). In a way, $\mathscr{X}$ and $\mathscr{Y}$ are equivalent: if we assume $\mathbf{r}^* \in \mathscr{X}$, then the only tuple that can be concluded to be in $\mathbf{r}^*$ is ab, and the same is true for $\mathscr{Y}$.

However, $\mathscr{X}$ provides the information that $a' \in \pi_A(\mathbf{r}^*)$, whereas $\mathscr{Y}$ does not. Hence $\mathscr{X}$ and $\mathscr{Y}$ are not "equivalent with respect to projection." □

Example 3.2. Let $\mathscr{X}$ consist of all those relations that contain at least one of the following two relations:

$$\begin{array}{|ccc|}\hline a & b & c \end{array}, \qquad \begin{array}{|ccc|}\hline a & b' & c' \\ a' & b & c' \end{array},$$

and let $\mathscr{Y}$ be defined in the same way by relations

$$\begin{array}{|ccc|}\hline a & b & c \end{array}, \qquad \begin{array}{|ccc|}\hline a & b' & c' \\ a' & b & c'' \end{array}.$$

It is easy to see that $\mathscr{X}$ and $\mathscr{Y}$ are equivalent with respect to projection, in the sense of the previous example. However, $\mathbf{r}^* \in \mathscr{X}$ implies

$$ab \in \pi_{AB}(\pi_{AC}(\mathbf{r}^*) \bowtie \pi_{BC}(\mathbf{r}^*)),$$

whereas this conclusion cannot be made under the assumption that $\mathbf{r}^* \in \mathscr{Y}$. Hence, $\mathscr{X}$ and $\mathscr{Y}$ are not equivalent "with respect to PJ-expressions." □

Example 3.3. Let $X = ABC$ and let

$$\mathscr{F} = \{r \in \mathscr{R}(X) : abc \in r \vee ab'c \in r\}.$$

Define

$$\mathscr{X} = \{\langle r, r\rangle : r \in \mathscr{F}\},$$
$$\mathscr{Y} = \{\langle r, s\rangle : r, s \in \mathscr{F}\}.$$

It is not difficult to prove that $\mathscr{X}$ and $\mathscr{Y}$ are equivalent with respect to all PS-expressions (because PS-expressions involve only unary operators). However, they are not equivalent with respect to PJ-expressions. Indeed, $\langle r^*, s^*\rangle \in \mathscr{X}$ implies

$$ac \in \pi_{AC}(r^* \bowtie s^*)$$

(notice that $r^* \bowtie s^* = r^* \cap r^* = r^*$), whereas this conclusion cannot be made under the assumption that $\langle r^*, s^*\rangle \in \mathscr{Y}$. □

The above examples suggest the following definition. For a relational multi-expression $\mathbf{f}$ and $\mathscr{X} \in \mathscr{I}$ ($\alpha(\mathbf{f}) = \alpha(\mathscr{X})$) define the $\mathbf{f}$-*information* in $\mathscr{X}$, denoted by $\mathscr{X}^{\mathbf{f}}$, to be

$$\mathscr{X}^{\mathbf{f}} = \bigcap \mathbf{f}(\mathscr{X})$$

(if $\mathbf{f} = \langle f_1, \ldots, f_k\rangle$, then $\bigcap \mathbf{f}(\mathscr{X})$ is understood as $\langle \bigcap f_1(\mathscr{X}), \ldots, \bigcap f_k(\mathscr{X})\rangle$). In other words, $\mathscr{X}^{\mathbf{f}}$ is the largest multirelation $\mathbf{s}$ such that $\mathbf{s} \subseteq \mathbf{f}(\mathbf{r}^*)$ for all $\mathbf{r}^* \in \mathscr{X}$. Putting it still another way, if $\mathscr{X}^{\mathbf{f}} = \langle s_1, \ldots, s_k\rangle$, then $t \in s_i$ means that from $\mathbf{r}^* \in \mathscr{X}$ we may conclude that $t \in f_i(\mathbf{r}^*)$.

Let $\mathscr{X}, \mathscr{Y} \in \mathscr{I}, \alpha(\mathscr{X}) = \alpha(\mathscr{Y})$. We say that $\mathscr{X}$ and $\mathscr{Y}$ are Ω-*equivalent* (in symbols $\mathscr{X} \equiv_\Omega \mathscr{Y}$) if $\mathscr{X}^{\mathbf{f}} = \mathscr{Y}^{\mathbf{f}}$ for any Ω-expression $\mathbf{f}$ with $\alpha(\mathbf{f}) = \alpha(\mathscr{X})$. Note that in Example 3.1, $\mathscr{X} \equiv_\varnothing \mathscr{Y}$ (we consider expressions of the form $f(R_1, \ldots, R_n)$

$= R_i$ to be the only relational $\varnothing$-expressions), but $\mathscr{X} \not\equiv_{\mathrm{P}} \mathscr{Y}$; in Example 3.2, $\mathscr{X} \equiv_{\mathrm{P}} \mathscr{Y}$, but $\mathscr{X} \not\equiv_{\mathrm{PJ}} \mathscr{Y}$; in Example 3.3, $\mathscr{X} \equiv_{\mathrm{PS}} \mathscr{Y}$, but $\mathscr{X} \not\equiv_{\mathrm{PJ}} \mathscr{Y}$.

A multitable **T** is said to Ω-*represent* $\mathscr{X}$ if Rep(**T**) $\equiv_\Omega \mathscr{X}$.

We are now ready to complete the definition of a representation system. A triple $\langle \mathscr{T}, \mathrm{Rep}, \Omega \rangle$, where $\mathscr{T}$, Rep, Ω are as described before, is a representation system if, for any Ω-expression **f** and for any multitable $\mathbf{T} \in \mathscr{T}$ ($\alpha(\mathbf{f}) = \alpha(\mathbf{T})$), there is a multitable $\mathbf{U} \in \mathscr{T}$ that Ω-represents **f**(Rep(**T**))—in other words, if **f**(**T**) can be defined for every Ω-expression **f** and multitable $\mathbf{T} \in \mathscr{T}$ ($\alpha(\mathbf{f}) = \alpha(\mathbf{T})$) in such a way that

$$\mathrm{Rep}(\mathbf{f}(\mathbf{T})) \equiv_\Omega \mathbf{f}(\mathrm{Rep}(\mathbf{T})). \tag{3.2}$$

Two multitables **T**, **U** are called Ω-*equivalent* if Rep(**T**) $\equiv_\Omega$ Rep(**U**) and *Rep-equivalent* if Rep(**T**) = Rep(**U**). Notice that for a given **T** and **f** there may exist many Ω-equivalent multitables $\mathbf{U} \in \mathscr{T}$ that Ω-represent **f**(Rep(**T**)).

There are two more conditions satisfied by most representation systems considered in this paper (these conditions are not part of the formal definition of a representation system). The first one is that any multirelation is—or at least can be identified with—a multitable, so that we may assume $\mathscr{R} \subseteq \mathscr{T}$. This reflects the fact that we are interested only in systems that extend the usual relational algebra. The second assumption (not valid in Section 9) is that for any multitable **T** and for any $\mathbf{r}, \mathbf{s} \in \mathscr{R}$

$$\mathbf{r} \in \mathrm{Rep}(\mathbf{T}) \wedge \mathbf{s} \supseteq \mathbf{r} \Rightarrow \mathbf{s} \in \mathrm{Rep}(\mathbf{T}). \tag{3.3}$$

This condition—which is essentially the *open world assumption* of Reiter [26]—is equivalent to saying that we are not able to represent negative information; that is, the knowledge that a relationship expressed by a certain tuple t is definitely *not* true in the real world $\mathbf{r}^*$ (or in $\mathbf{f}(\mathbf{r}^*)$, where f is some relational expression). An intuitive consequence of this assumption is that for any $\mathbf{r} \in \mathscr{R}$

$$\mathrm{Rep}(\mathbf{r}) = \{\mathbf{s} \in \mathscr{R} : \mathbf{s} \supseteq \mathbf{r}\}.$$

We conclude this section by the following simple lemma.

LEMMA 3.1. *If Ω contains only unary operators, then $\langle \mathscr{T}, Rep, \Omega \rangle$ is a representation system iff $f(T)$ can be defined for every relational (rather than multirelational) Ω-expression f and every $T \in \mathscr{T}$ with $\alpha(T) = \alpha(f)$, in such a way that $Rep(f(T)) \equiv_\Omega f(Rep(T))$.*

The easy proof is omitted.

4. *Codd Tables*

In this section we prove that the usual tables with null values @ (see [7]) provide a basis for a representation system with Ω = PS. We also show that these tables can support neither Ω = PSU nor Ω = PJ.

By a *@-tuple* on X we mean any function t that associates a value $t(A) \in D(A) \cup \{@\}$ with every $A \in X$. A *Codd table* on X is any finite set of @-tuples on X, and a *Codd multitable* of type $\langle X_1, \ldots, X_n \rangle$ is any sequence $\mathbf{T} = \langle T_1, \ldots, T_n \rangle$ where T_i is a Codd table on X_i, $1 \le i \le n$. The set of all Codd multitables is denoted by $\mathscr{T}_@$.

Let us define a partial order $\le$ on $D \cup \{@\}$ in such a way that $@ < a$, $a \in D$ are the only nontrivial relationships. For two @-tuples t, t' on X, we write $t \le t'$

if $t(A) \leq t'(A)$ for every $A \in X$, that is, if t and t' agree on every $A \in X$ such that $t(A) \neq @$. For any Codd table T on X we define

$$\mathrm{Rep}(T) = \{r \in \mathscr{R}(X) : \text{for every } t \in T \text{ there is } t' \in r \text{ such that } t \leq t'\},$$

and for every Codd multitable $\mathbf{T} = \langle T_1, \ldots, T_n \rangle$,

$$\mathrm{Rep}(\mathbf{T}) = \mathrm{Rep}(T_1) \times \cdots \times \mathrm{Rep}(T_n).$$

Intuitively, $\mathbf{r} \in \mathrm{Rep}(\mathbf{T})$ iff $\mathbf{r}$ contains a multirelation obtained from $\mathbf{T}$ by replacing all occurrences of @ by some values in appropriate attribute domains (different occurrences may be replaced by different values, but equal values are allowed as well).

Let us now have a closer look at the notion of Ω-equivalence of homogeneous sets of multirelations for some simple cases of Ω.

We first note the following trivial fact:

$$\mathscr{X} \equiv_{\varnothing} \mathscr{Y} \Leftrightarrow \bigcap \mathscr{X} = \bigcap \mathscr{Y}. \tag{4.1}$$

Let us now consider the case of $\Omega = \mathrm{P}$.

THEOREM 4.1. *Any $\mathscr{X} \in \mathscr{I}$ can be P-represented by a Codd multitable.*

PROOF. Let us begin with the simple case where $\alpha(\mathscr{X}) = X$. For any $Y \subseteq X$, let $Q(Y)$ be obtained from $\mathscr{X}^{\pi_Y}$ by extending with @'s every tuple $t \in \mathscr{X}^{\pi_Y}$ to a @-tuple $\bar{t}$ on X (notice that $Q(\varnothing) = \varnothing$ if $\varnothing \in \mathscr{X}$ and $Q(\varnothing)$ consists of a tuple of @, otherwise). Define $T = \bigcup_{Y \subseteq X} Q(Y)$. We shall prove that $\mathrm{Rep}(T) \equiv_{\mathrm{P}} \mathscr{X}$. Indeed, consider an arbitrary $Y \subseteq X$. If $t \in \mathscr{X}^{\pi_Y}$, then clearly T contains a @-tuple t' that agrees with t on Y, and consequently every $r \in \mathrm{Rep}(T)$ contains a tuple u that agrees with t on Y; that is, $t \in \mathrm{Rep}(T)^{\pi_Y}$. Conversely, assume that $t \notin \mathscr{X}^{\pi_Y}$, $\alpha(t) = Y$. Then no @-tuple in T agrees with t on Y, and consequently there exists a relation $r \in \mathrm{Rep}(T)$ not containing any tuple agreeing with t on Y (r may be obtained by replacing every null value in T by a value in the corresponding attribute domain, in such a way that if $A \in Y$ then the value replacing an occurrence of @ in column A of T is different from $t(A)$). Hence $t \notin \mathrm{Rep}(T)^{\pi_Y}$, which concludes the proof that $\mathrm{Rep}(T) \equiv_{\mathrm{P}} \mathscr{X}$.

In the general case, in which $\alpha(\mathscr{X}) = \langle X_1, \ldots, X_n \rangle$, we construct tables T_i such that $\mathrm{Rep}(T_i) \equiv_{\mathrm{P}} \mathrm{pr}_i(\mathscr{X})$, $1 \leq i \leq n$. It is then easily seen that $\mathbf{T} = \langle T_1, \ldots, T_n \rangle$ P-represents $\mathscr{X}$. □

Let us note the following simple fact concerning the P-equivalence of Codd multitables.

THEOREM 4.2. *Any two P-equivalent Codd multitables are Rep-equivalent.*

In other words, if $\mathrm{P} \in \Omega$, then $\mathbf{T} \equiv_{\Omega} \mathbf{U}$ iff $\mathbf{T} \equiv \mathbf{U}$. The easy proof is omitted.

Clearly, two Codd multitables $\langle T_1, \ldots, T_n \rangle$, $\langle U_1, \ldots, U_n \rangle$ of the same type are Rep-equivalent iff $T_i \equiv U_i$, $1 \leq i \leq n$. The following structural characterization of Rep-equivalence of Codd tables was proved by Biskup [3]:

THEOREM 4.3. *For any Codd tables T, U of the same type $T \equiv U$ iff for every $t \in T$ there is $u \in U$ such that $t \leq u$ and for every $u \in U$ there is $t \in T$ such that $u \leq t$.*

The S-equivalence is even simpler than the P-equivalence, since it turns out to coincide with the $\varnothing$-equivalence (see (4.1)). Before we prove this, notice that $\sigma_E(r)$ may be expressed as $r \cap \sigma_E(\mathbb{1}_Y)$, where $Y = \alpha(r)$ and $\mathbb{1}_Y = \times_{A \in Y} D(A)$, and this is true for arbitrary conditions E: $\mathbb{1}_Y \rightarrow$ {**true**, **false**}, not only for those generated by atomic conditions of the form $(A = a)$, $(A = B)$.

THEOREM 4.4. *For any $\mathscr{X}$, $\mathscr{Y} \in \mathscr{I}$ of the same type,*

$$\mathscr{X} \equiv_S \mathscr{Y} \Leftrightarrow \mathscr{X} \equiv_\varnothing \mathscr{Y}.$$

PROOF. Let $\mathscr{X}$, $\mathscr{Y} \in \mathscr{I}$, and let $f(R_1, \ldots, R_n) = \sigma_E(R_i)$ be an arbitrary relational S-expression ($\alpha(\mathscr{X}) = \alpha(\mathscr{Y}) = \alpha(f)$). If $\mathscr{X} \equiv_\varnothing \mathscr{Y}$ then, by (4.1), $\bigcap \mathscr{X} = \bigcap \mathscr{Y}$, which implies $\bigcap \mathrm{pr}_i(\mathscr{X}) = \bigcap \mathrm{pr}_i(\mathscr{Y})$. Hence

$$\begin{aligned}\mathscr{X}^f &= \bigcap_{r \in \mathrm{pr}_i(\mathscr{X})} \sigma_E(r) = \bigcap_{r \in \mathrm{pr}_i(\mathscr{X})} (r \cap \sigma_E(\mathbb{1}_{\alpha(r)})) = \sigma_E(\mathbb{1}_{\alpha(r)}) \cap \bigcap \mathrm{pr}_i(\mathscr{X})\\ &= \sigma_E(\mathbb{1}_{\alpha(r)}) \cap \bigcap \mathrm{pr}_i(\mathscr{Y}) = \mathscr{Y}^f,\end{aligned}$$

which easily implies $\mathscr{X} \equiv_S \mathscr{Y}$. □

Notice that the S-equivalence coincides with the $\varnothing$-equivalence even if we allow arbitrary functions E: $\mathbb{1}_Y \rightarrow$ {**true**, **false**} as selection conditions. Clearly, by Theorem 4.4, $\mathscr{X} \equiv_P \mathscr{Y}$ implies $\mathscr{X} \equiv_S \mathscr{Y}$. However, the PS-equivalence does not coincide with the P-equivalence on $\mathscr{I}$, as shown by the following example.

Example 4.1. Let

$$\mathscr{X} = \{\boxed{abc'}, \boxed{ab'c}\}, \qquad \mathscr{Y} = \{\boxed{abc}, \boxed{ab'c'}\}.$$

We have $\mathscr{X} \equiv_P \mathscr{Y}$ since both $\mathscr{X}$ and $\mathscr{Y}$ can be P-represented by $T = \boxed{a@@}$. Consequently, also $\mathscr{X} \equiv_S \mathscr{Y}$. But $\mathscr{X} \not\equiv_{PS} \mathscr{Y}$, since for the PS-expression

$$f(R) = \pi_A(\sigma_{(B=b)\vee(C=c)}(R))$$

we have

$$\mathscr{X}^f = \{a\} \neq \varnothing = \mathscr{Y}^f.$$

In other words, $\mathscr{X}$ and $\mathscr{Y}$ are not distinguishable by either P-expressions or S-expressions, but *are* distinguishable by PS-expressions. □

Before we prove that $\langle \mathscr{T}_@, \mathrm{Rep}, \mathrm{PS} \rangle$ is indeed a representation system, let us note the following two simple facts:

LEMMA 4.1. *Every PSUJR-expression* **f** *is monotone; that is,* $\mathbf{r} \subseteq \mathbf{s} \Rightarrow \mathbf{f}(\mathbf{r}) \subseteq \mathbf{f}(\mathbf{s})$.

The easy proof, by induction on the number of operators in **f**, is omitted.

Let $\mathscr{X}$, $\mathscr{Y} \in \mathscr{I}$, $\alpha(\mathscr{X}) = \alpha(\mathscr{Y})$. We say that $\mathscr{X}$ and $\mathscr{Y}$ are *coinitial* (in symbols, $\mathscr{X} \approx \mathscr{Y}$) if for any $\mathbf{r} \in \mathscr{X}$ there is $\mathbf{s} \in \mathscr{Y}$ such that $\mathbf{s} \subseteq \mathbf{r}$, and for any $\mathbf{s} \in \mathscr{Y}$ there is $\mathbf{r} \in \mathscr{X}$ such that $\mathbf{r} \subseteq \mathbf{s}$. Obviously, for any monotone **f**

$$\mathscr{X} \approx \mathscr{Y} \Rightarrow \mathbf{f}(\mathscr{X}) \approx \mathbf{f}(\mathscr{Y}). \tag{4.2}$$

LEMMA 4.2. *If $\mathscr{X}$ and $\mathscr{Y}$ are coinitial, then $\mathscr{X} \equiv_{PSUJR} \mathscr{Y}$.*

PROOF. Let $\mathscr{X} \approx \mathscr{Y}$. Then for every $\mathbf{r} \in \mathscr{X}$ there is a $\varphi(\mathbf{r}) \in \mathscr{Y}$ such that $\varphi(\mathbf{r}) \subseteq \mathbf{r}$. By using the previous lemma, for any PSUJR-expression **f** we have

$$\mathscr{X}^{\mathbf{f}} = \bigcap_{\mathbf{r} \in \mathscr{X}} \mathbf{f}(\mathbf{r}) \supseteq \bigcap_{\mathbf{r} \in \mathscr{X}} \mathbf{f}(\varphi(\mathbf{r})) \supseteq \bigcap_{\mathbf{s} \in \mathscr{Y}} \mathbf{f}(\mathbf{s}) = \mathscr{Y}^{\mathbf{f}},$$

and similarly $\mathscr{Y}^{\mathbf{f}} \supseteq \mathscr{X}^{\mathbf{f}}$. □

We are now ready to prove the main result of this section.

THEOREM 4.5. *It is possible to correctly evaluate PS-expressions over Codd tables; more formally, $\langle \mathcal{T}_@, Rep, PS\rangle$ is a representation system.*

PROOF. By Lemma 3.1, it is sufficient to show that for any $\mathbf{T} \in \mathcal{T}_@$ and any relational PS-expression f with $\alpha(f) = \alpha(T)$ it is possible to define $f(\mathbf{T})$ in such a way that $\mathrm{Rep}(f(\mathbf{T})) \equiv_{PS} f(\mathrm{Rep}(\mathbf{T}))$.

We define $f(\mathbf{T})$ inductively, by using the following rules:

$$\text{If} \quad \mathbf{T} = \langle T_1, \ldots, T_n\rangle, \quad \text{then} \quad \mathrm{pr}_i(\mathbf{T}) = T_i,$$
$$\pi_Y(T) = \{t[Y] : t \in T\},$$
$$\sigma_E(T) = \{t \in T : E_*(t) = \mathbf{true}\},$$

where

$$E_*(t) = \begin{cases} \mathbf{true} & \text{if } E(u) \text{ for every } u \in \mathrm{Compl}(t), \\ \mathbf{false} & \text{otherwise,} \end{cases}$$

and $\mathrm{Compl}(t)$ denotes the set of all tuples u (not containing null values) such that $t \leq u$.

By Lemma 4.2, it is now sufficient to prove that

$$\mathrm{Rep}(f(\mathbf{T})) \approx f(\mathrm{Rep}(\mathbf{T})) \tag{4.3}$$

for all f and $\mathbf{T}$. Let us first notice that

$$\text{if} \quad \mathbf{T} = \langle T_1, \ldots, T_n\rangle, \quad \text{then} \quad \mathrm{Rep}(\mathrm{pr}_i(\mathbf{T})) = \mathrm{pr}_i(\mathrm{Rep}(\mathbf{T})), \tag{4.4}$$

and that for any Codd table T

$$\mathrm{Rep}(\pi_Y(T)) = \pi_Y(\mathrm{Rep}(T)), \tag{4.5}$$
$$\mathrm{Rep}(\sigma_E(T)) \approx \sigma_E(\mathrm{Rep}(T)). \tag{4.6}$$

We show (4.6) ((4.4) and (4.5) are obvious).

Let $r \in \mathrm{Rep}(\sigma_E(T))$. Then r contains a relation $s \in \sigma_E(\mathrm{Rep}(T))$ obtained by replacing nulls in the @-tuples $t \in T$ that belong to $\sigma_E(T)$ in such a way that the resulting tuples are in r, and by replacing nulls in every $t \in T\backslash\sigma_E(T)$ so as to obtain a tuple t^* with $E(t^*) = \mathbf{false}$ (this is possible since for every $t \in T\backslash\sigma_E(T)$, $E_*(t) = \mathbf{false}$).

Conversely, let $s \in \sigma_E(\mathrm{Rep}(T))$. Then s contains a relation $r \in \mathrm{Rep}(\sigma_E(T))$ obtained from T by replacing those @-tuples in T that belong to $\sigma_E(T)$ so as to obtain tuples in s, and by omitting the remaining @-tuples.

The desired formula (4.3) now easily follows by induction on the number of operators in f. We show, as an example, the inductive step in the case in which $f = \sigma_E g$, under the inductive assumption $\mathrm{Rep}(g(\mathbf{T})) \approx g(\mathrm{Rep}(\mathbf{T}))$:

$$\begin{aligned} \mathrm{Rep}(f(\mathbf{T})) &= \mathrm{Rep}(\sigma_E(g(\mathbf{T}))) \approx \sigma_E(\mathrm{Rep}(g(\mathbf{T}))) \\ &\approx \sigma_E(g(\mathrm{Rep}(\mathbf{T}))) = f(\mathrm{Rep}(\mathbf{T})) \end{aligned}$$

(we made use of (4.6) and then of (4.2)). □

Note that in Theorem 4.5 selection can be arbitrary, on the basis of an arbitrary function from tuples to {**true, false**}. This includes as a special case selection based on arbitrary Boolean combinations generated by atomic conditions of the form $(A \leq a)$, $(A \leq B)$ $(A, B \in \mathcal{U}$, $a \in D(A)$; we assume that the attribute domains are linearly ordered). These cases are treated in more detail in the next section.

The next two theorems say that, roughly speaking, our representation system based on Codd tables cannot, in addition, handle either union or join.

THEOREM 4.6. *It is not possible to correctly evaluate PSU-expressions over Codd tables; more formally* $\langle \mathcal{T}_@, \mathit{Rep}, \mathit{PSU}\rangle$ *is not a representation system.*

PROOF. We now give an example of a PSU-expression $\mathbf{f}$ and a Codd table T such that $\mathbf{f}(\mathrm{Rep}(T))$ is not PSU-representable by any Codd multitable. Let

$$\mathbf{f}(R) = \langle \sigma_{A=a}(R), \sigma_{A\neq a}(R)\rangle \qquad (\alpha(\mathbf{f}) = AB,\ a \in D(A)),$$

and let $T = \boxed{@b}$. Suppose that $\mathbf{U} = \langle U_1, U_2\rangle$ PSU-represents $\mathbf{f}(\mathrm{Rep}(T))$. Then obviously $\mathbf{U}$ P-represents $\mathbf{f}(\mathrm{Rep}(T))$. But the multitable $\langle \varnothing, \varnothing\rangle$ is easily seen to P-represent $\mathbf{f}(\mathrm{Rep}(T))$ (see the construction in the proof of Theorem 4.1), so that by Theorems 4.2 and 4.3, $\mathbf{U} = \langle \varnothing, \varnothing\rangle$.

Consider the relational expression $g(S_1, S_2) = \pi_B(S_1 \cup S_2)$. We have

$$\mathbf{f}(\mathrm{Rep}(T))^g = \bigcap g\mathbf{f}(\mathrm{Rep}(T)) = \bigcap \pi_B(\mathrm{Rep}(T)) = \{b\} \neq \varnothing = \mathrm{Rep}(\mathbf{U})^g,$$

that is, $\mathbf{U}$ does not PSU-represent $\mathbf{f}(\mathrm{Rep}(T))$, contrary to our assumption. □

It may be noted that if we restrict ourselves to the case in which all relational PSU-expressions considered involve only one relation symbol, then $f(\mathrm{Rep}(T))$ can always be PSU-represented by a Codd table U (see [13]). This is a simple consequence of the fact that any PSU-expression f involving only one relation symbol can be transformed into an equivalent PS-expression of the form $\pi_Y(\sigma_E(R))$. The easy proof of this fact, by induction on the number of operators in f, is omitted.

THEOREM 4.7. *It is not possible to correctly evaluate PJ-expressions over Codd tables; more formally,* $\langle \mathcal{T}_@, \mathit{Rep}, \mathit{PJ}\rangle$ *is not a representation system.*

PROOF. We give an example of a Codd table T and a PJ-expression f such that $f(\mathrm{Rep}(T))$ is not PJ-representable by any Codd table. Let

$$T = \{a@c,\ a'@c'\} \qquad (a \neq a',\ c \neq c')$$
$$f(R) = \pi_{AC}(R) \bowtie \pi_B(R).$$

It is easy to see that $f(\mathrm{Rep}(T)) \equiv_{\mathrm{P}} \mathrm{Rep}(T)$. By Theorem 4.2, if there is a Codd table U PJ-representing $f(\mathrm{Rep}(T))$, then $T \equiv U$, and consequently $f(\mathrm{Rep}(T)) \equiv_{\mathrm{PJ}} \mathrm{Rep}(T)$.

This last equivalence is, however, not true. Indeed, let

$$g(R) = \pi_{AC}(\pi_{AB}(R) \bowtie \pi_{BC}(R)).$$

We have

$$\mathrm{Rep}(T)^g = \left|\begin{array}{cc} a & c \\ a' & c' \end{array}\right|,$$

while

$$f(\mathrm{Rep}(T))^g = \left|\begin{array}{cc} a & c \\ a' & c' \\ a & c' \\ a' & c \end{array}\right|. \qquad □$$

A representation system with $\Omega \supseteq \mathrm{PJ}$ is considered in Section 6.

Now we compare our approach to that of Biskup [3]. Roughly speaking, Biskup defines the union of two Codd tables to be the usual set-theoretical union, so that

$$\mathrm{Rep}(T \cup U) = \{r \cup s : r \in \mathrm{Rep}(T) \wedge s \in \mathrm{Rep}(U)\},$$

and he defines the join of two Codd tables in such a way that

$$\mathrm{Rep}(T \bowtie U) \equiv_{\mathrm{P}} \{r \bowtie s : r \in \mathrm{Rep}(T) \wedge s \in \mathrm{Rep}(U)\}. \tag{4.7}$$

His definition of join is based on the following informal matching rule:

$$@ \neq @, \qquad @ \neq a.$$

(In fact, it may be shown that the sets on both sides of (4.7) are coinitial so that they are PSUJ-equivalent.)

These definitions of union and join are semantically correct only if we assume that both arguments are independent; that is, every substitution of values in appropriate attribute domains for the null values in T and U is meaningful. This is clearly not the case when we want to correctly evaluate relational expressions such as

$$f(R) = \sigma_{A=a}(R) \cup \sigma_{A\neq a}(R),$$
$$g(R) = \pi_{AB}(R) \bowtie \pi_{AC}(R),$$

over a Codd table. Consequently, Biskup's approach does not generalize to arbitrary relational expressions, though in a sense, it does give correct results for single relational operators. On the other hand, Biskup's approach is more general in that it allows "universal" null values in addition to the usual "existential" null values @.

5. *Evaluating a Selection over a Codd Table*

Recall that the selection of a Codd table is defined by

$$\sigma_E(T) = \{t \in T : E_*(t) = \mathbf{true}\},$$

where $E_*(t) = \mathbf{true}$ iff $E(t') = \mathbf{true}$ for every $t' \in \mathrm{Compl}(t)$. Of course, we should have a more efficient method of computing $E_*(t)$ than that involving the computation of $E(t')$ for every $t' \in \mathrm{Compl}(t)$. ($\mathrm{Compl}(t)$ may be infinite if attribute domains are infinite.)

In this section we always assume that the selection condition E is of the form of an arbitrary Boolean combination (using $\neg$, $\vee$, $\wedge$) of certain atomic conditions. The case in which these conditions are of the form (A IN F), $F \subseteq D(A)$ and in which subset entries of the form G, $G \subseteq D(A)$, meaning "an unknown value in G," are allowed in the table, instead of just @, "an unknown value in $D(A)$," was extensively studied in [19], [20], and [21].

Here we sketch a method of computing $E_*(t)$ in the case in which the atomic conditions are of the form $(A = a)$, $(A = B)$, $(A \leq a)$, $(A \leq B)$ ($A, B \in \mathcal{U}$, $a \in D(A)$; in the third and fourth cases we assume a linear order on the corresponding attribute domains). The conditions $(A = a)$, $(A \leq a)$ are called *unary*, whereas $(A = B)$, $(A \leq B)$ are called *binary*. For simplicity, throughout this section we assume that every attribute domain is the same set D.

Note that we may "precompute" $E_*(t)$ by substituting in E the value $t(A)$ for every occurrence of A, for every A such that $t(A) \neq @$. This has the effect that some of the binary conditions become unary, and some conditions are reduced to $(a = b)$ or $(a \leq b)$, which can be replaced by **true** or **false** and then eliminated by using the absorption-type Boolean axioms. Let E' be the resulting condition. Clearly, the whole process can be carried out in time linear in the length of E. In this way our problem is reduced to evaluating $E'_*(t')$ where t' is a tuple of null values. This is obviously equivalent to testing whether E' is a tautology.

Let us begin with the case involving only conditions of the form $(A = a)$, $(A = B)$. We transform E' into a conjunctive normal form $\bigwedge_i \bigvee_j E_{ij}^{\epsilon_{ij}}$, where E_{ij} is an atomic condition, and $E_{ij}^{\epsilon_{ij}}$ denotes E_{ij} or $\neg E_{ij}$, depending on whether ϵ_{ij} is 1 or 0, respectively; we may also assume that no disjunct of this normal form contains both an equality and the negation to this equality. E' is a tautology iff for every i, $\bigvee_j E_{ij}^{\epsilon_{ij}}$ is a tautology. A disjunction of the form $\bigvee_j E_j^{\epsilon_j}$ is a tautology iff $\bigwedge_j E_j^{1-\epsilon_j}$ is not satisfiable. The satisfiability of this last conjunction can be tested in the following way. We construct a (nondirected) graph with vertices corresponding to attributes and constants occurring in $\bigwedge_j E_j^{1-\epsilon_j}$ (different occurrences of the same constant or attribute correspond to the same vertex), two vertices x, y joined by an edge iff the (nonnegated) condition $(x = y)$ appears in our conjunction.

It is clear that the conjunction is satisfiable iff every connected component of our graph contains at most one vertex corresponding to a constant. Obviously, finding the connected components can be done in time linear with respect to the number of vertices and edges of our graph, that is, $O(n)$, where n is the length of our conjunction (see, e.g., [11]). (Identifying *different* occurrences of attributes and constants may require $\Omega(n \log n)$ time.) The most complex part of this algorithm is, however, the transformation into a conjunctive normal form, since in the worst case it may involve an exponential growth of the length of our expression. We should not expect a substantially more efficient algorithm to evaluate $E_*(t)$, since it obviously contains as a special case the problem of deciding whether a Boolean expression is a tautology, a problem that is known to be NP-hard (see, e.g., [9]).

We now consider the case in which all four types of atomic conditions are allowed. Since

$$(x = y) \Leftrightarrow (x \leq y) \wedge (y \leq x), \tag{5.1}$$

our problem is reduced, in a similar way as before, to testing for the satisfiability of a conjunction $\bigwedge_j E_j$, where each E_j is either of the form $(x \leq y)$ or $(x < y)$. (It may be noted, however, that using (5.1) to eliminate equality may be not the most efficient approach.) Let us construct a directed graph with vertices corresponding to different attributes occurring in $\bigwedge_j E_j$, with an $<$-edge or $\leq$-edge $\langle x, y\rangle$ from x to y iff $(x < y)$ or $(x \leq y)$, respectively, occurs in our conjunction. We associate with every vertex x an interval $I(x)$ with the beginning

$$b(x) = \max\{y \in D : (y < x) \text{ or } (y \leq x) \text{ appears in } \bigwedge_j E_j\}$$

and end

$$e(x) = \min\{y \in D : (x < y) \text{ or } (x \leq y) \text{ appears in } \bigwedge_j E_j\}$$

($b(x)$ is included into $I(x)$ iff $(b(x) < x)$ appears in the conjunction; similarly for $e(x)$).

We find strongly connected components of the graph consisting of $\leq$-edges (see, e.g., [25]; x and y are in the same strongly connected component iff there is a path from x to y and a path from y to x).

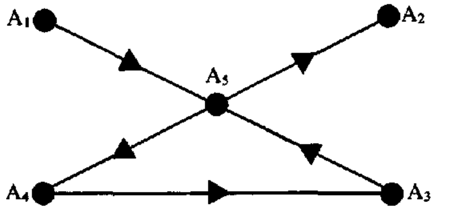
For every strongly connected component, we identify all vertices of the component and replace them with a single vertex x_C with $I(x_C) = \bigcap_{y \in C} I(y)$, where C is the set of vertices of the component. (In the resulting graph, there is an edge from $x_{C'}$ to $x_{C''}$, $C' \neq C''$, iff there was an edge from some $x' \in C'$ to some $x'' \in C''$ in the original graph.) The graph obtained is always acyclic and we test for the satisfiability of our conjunction in the following way. We try to assign a value $v(x) \in I(x)$ to every vertex x in such a way that $v(x) < v(y)(v(x) \leq v(y))$ if there is a $<$-edge ($\leq$-edge, respectively) $\langle x, y\rangle$. Let us assume, for simplicity, that our graph contains only ≤-edges, and that every $I(x)$ contains both $b(x)$ and $e(x)$ (the general case can be treated in a similar way). We set initially $v(x) = b(x)$ for any x, and we perform a breadth-first search (see, e.g., [11]) starting from the set of sources (i.e., vertices with no incoming edges). For any edge $\langle x, y\rangle$ processed by the search we set $v(y) := \max(v(y), v(x))$. If at any stage of the process this results in $v(y) > e(y)$, then clearly our conjunction is not satisfiable. Otherwise, the process determines an assignment of values to attributes that makes our conjunction true. We leave the details to the reader.

Exa.nple 5.1. Let our conjunction be

$$(A_1 \le 50) \wedge (A_4 \ge 100) \wedge (A_3 \le 200) \wedge (A_1 \le A_5)$$
$$\wedge (A_5 \le A_2) \wedge (A_5 \le A_4) \wedge (A_4 \le A_3) \wedge (A_3 \le A_5)$$

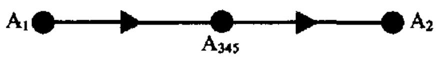
(the domain of each attribute is the set of integers).

Our graph has the form



$I(A_1) = (-\infty, 50]$,
$I(A_2) = (-\infty, \infty)$,
$I(A_3) = (-\infty, 200]$,
$I(A_4) = [100, \infty)$,
$I(A_5) = (-\infty, \infty)$.

The strongly connected components have vertex sets $\{A_1\}$, $\{A_2\}$, $\{A_3, A_4, A_5\}$, and our graph is transformed into



$I(A_{345}) = [100, 200]$.

The breadth-first search produces $v(A_1) = -\infty$, $v(A_{345}) = 100$, $v(A_2) = 100$, which means that our conjunction is satisfiable. □

6. *V-Tables*

One of the reasons for the inability of Codd tables to correctly support the join operation appears to be the fact that we are not able to represent the information that two different occurrences of @ represent the same value. In this section we consider tables where, for any $A \in \mathscr{U}$, we have an infinite set of possible "null values." The interpretation of such a table is that the values represented by two occurrences of the same null value, though unknown, are the same. Such tables will turn out to be suitable for a representation system with $\Omega = \mathrm{PS^+UJ}$.

Formally, for every $A \in \mathscr{U}$, let $V(A)$ be a countably infinite set of symbols called *variables*. We assume that $V(A) \cap D = \varnothing$, $V(A) \cap V(B) = \varnothing$ if $D(A) \neq D(B)$, and $V(A) = V(B)$ if $D(A) = D(B)$. By a *V-tuple* on X, we mean any mapping t that associates an element $t(A) \in D(A) \cup V(A)$ with every $A \in X$. A *V-table* on X is any finite set of V-tuples on X, and a *V-multitable* of type $\langle X_1, \ldots, X_n\rangle$ is any sequence $\mathbf{T} = \langle T_1, \ldots, T_n\rangle$ where T_i is a V-table on X_i, $1 \le i \le n$ (notice that the same variable may occur in several T_i's). The set of all V-multitables is denoted by $\mathscr{T}_V$.

Let $V = \bigcup_{A\in\mathscr{U}} V(A)$. By a *valuation*, we mean any mapping v: $V \rightarrow D$ such that $x \in V(A)$ implies $v(x) \in D(A)$. Any valuation can be extended to the set of constants by putting $v(c) = c$ for every $c \in D$; to V-tuples, by defining, for any V-tuple t on

X, $v(t)$ to be the tuple on X satisfying

$$(v(t))(A) = v(t(A)) \qquad \text{for every} \quad A \in X; \tag{6.1}$$

to V-tables,

$$v(T) = \{v(t) : t \in T\}; \tag{6.2}$$

and finally to V-multitables $\mathbf{T} = \langle T_1, \ldots, T_n \rangle$,

$$v(\mathbf{T}) = \langle v(T_1), \ldots, v(T_n) \rangle. \tag{6.3}$$

The intuition given at the beginning of this section can now be formalized by defining, for any V-multitable, $\mathbf{T} = \langle T_1, \ldots, T_n \rangle$ of type $\langle X_1, \ldots, X_n \rangle$:

$$\operatorname{Rep}(\mathbf{T}) = \{\mathbf{r} \in \mathscr{R}(X_1, \ldots, X_n) : \text{there exists a valuation } v \text{ such that } v(\mathbf{T}) \subseteq \mathbf{r}\}. \tag{6.4}$$

Note that in view of this definition, any Codd multitable can informally be identified with a V-multitable obtained by replacing any occurrence of @ by a different variable.

Let us associate with a V-multitable $\mathbf{T} = \langle T_1, \ldots, T_n \rangle$ a first-order formula $\phi(\mathbf{T})$ (of a many-sorted predicate calculus) defined as

$$\exists x_1 \cdots \exists x_m \bigwedge_t \phi(t)$$

where t ranges over all V-tuples appearing in $\mathbf{T}$ and, for $t \in T_i$, $\phi(t)$ is $R_i(t)$, where R_i is a predicate symbol of a suitable type and $x_1, \ldots, x_m$ are all variables occurring in $\mathbf{T}$. It is easy to see that $\operatorname{Rep}(\mathbf{T})$ is simply the set of all finite models of $\phi(\mathbf{T})$ (over a fixed universe given by $D(A)$, $A \in \mathscr{U}$).

In this section we always assume that all attribute domains are infinite.

The following is the main result of this paper.

THEOREM 6.1. *It is possible to correctly evaluate PS^+UJ-expressions over V-tables; more formally $\langle \mathscr{T}_V, Rep, PS^+UJ \rangle$ is a representation system.*

This theorem is proved in the next section, in which we develop a suitable formal notion, that of a *conditional table* (see also another proof in [23]). Here we only give the relevant definitions of the relational operators over V-tables.

For any V-multitable $\mathbf{T} = \langle T_1, \ldots, T_n \rangle$ we put

$$\operatorname{pr}_i(\mathbf{T}) = T_i, \qquad 1 \le i \le n. \tag{6.5}$$

The definition of projection is the "natural one,"

$$\pi_Y(T) = \{t[Y] : t \in T\}, \tag{6.6}$$

and so is the definition of join of two V-tables T, W on X, Z, respectively:

$$T \bowtie W = \{t : \alpha(t) = X \cup Z \wedge t[X] \in T \wedge t[Z] \in W\}. \tag{6.7}$$

The union operator acts on V-tables as the usual set-theoretical union.

Finally,

$$\sigma_E(T) = \{t \in T : E_*(t) = \textbf{true}\} \tag{6.8}$$

where

$$E_*(t) = \begin{cases} \textbf{true} & \text{if } E(v(t)) = \textbf{true} \text{ for every valuation } v, \\ \textbf{false} & \text{otherwise.} \end{cases}$$

Note that, if E is positive, then $E_*(t)$ can be computed in a very simple way by using the following rule: We evaluate every atomic condition $(A = B)$ in E to **true** if $t(A) = t(B)$ and to **false** otherwise; we evaluate every atomic condition $(A = a)$ to **true** if $t(A) = a$ and to **false** otherwise; and then we use $\vee$, $\wedge$ in the natural way. In other words,

$$(A = B)_*(t) = \begin{cases} \textbf{true} & \text{if } t(A) = t(B), \\ \textbf{false} & \text{otherwise}, \end{cases}$$
$$(A = a)_*(t) = \begin{cases} \textbf{true} & \text{if } t(A) = a, \\ \textbf{false} & \text{otherwise}, \end{cases}$$
$$(E \vee E')(t) = E_*(t) \vee E'_*(t),$$
$$(E \wedge E')(t) = E_*(t) \wedge E'_*(t).$$

The correctness of this rule follows easily from the fact that, if there is a valuation v such that $E(v(t)) = \textbf{false}$, then $E(v'(t)) = \textbf{false}$ for any valuation v' such that $v'(x) \neq v'(y)$ for $x \neq y$ and $v'(x)$ does not appear in E for any $x \in V$. Note that this rule does *not* evaluate $E_*(t)$ correctly if E contains negation, or if attribute domains are finite.

The above rules, together with the obvious rule

$$\mathbf{f}(\mathbf{T}) = \langle f_1(\mathbf{T}), \ldots, f_k(\mathbf{T})\rangle \tag{6.9}$$

for any $\mathbf{f} = \langle f_1, \ldots, f_k\rangle$, define inductively $\mathbf{f}(\mathbf{T})$ for any multirelational PS$^+$UJ-expression $\mathbf{f}$ and any V-multitable $\mathbf{T}$ ($\alpha(\mathbf{T}) = \alpha(\mathbf{f})$).

To sum up, we evaluate PS$^+$UJ-expressions over V-tables in *exactly the same way* as if the variables were values in the attribute domains. Let us emphasize that the fact that this simple method of evaluating PS$^+$UJ-expressions gives correct results is not quite trivial (see the proof in the next section).

Example 6.1. Let us evaluate

$$f(T) = \pi_{AC}(\pi_{AB}(T) \bowtie \sigma_{(B=C)\vee(C=c)}(\pi_{BC}(T))),$$

where T is the following V-table:

A	B	C
x	y	c
a	b	c
a'	b'	c'
a	y	z
x	d	d

We get

$$\begin{aligned} \pi_{AB}(T) &= \{xy, ab, a'b', xy, xd\}, \\ \pi_{BC}(T) &= \{yc, bc, b'c', yz, dd\}, \\ \sigma_{(B=C)\vee(C=c)}(\pi_{BC}(T)) &= \{yc, bc, dd\}, \\ \pi_{AB}(T) \bowtie \sigma_{(B=C)\vee(C=c)}(\pi_{BC}(T)) &= \{xyc, abc, xdd\}, \\ f(T) &= \{xc, ac, xd\}. \end{aligned}$$ □

Rather suprisingly, it turns out that V-multitables, which are "more powerful" than Codd multitables, cannot support a representation system with $\Omega = $ PS.

THEOREM 6.2. *It is not possible to correctly evaluate PS-expressions over V-tables; more formally, $\langle \mathcal{T}_V, Rep, PS\rangle$ is not a representation system.*

PROOF. Let T be the following V-table:

A	B	C
a	y	c
a'	y	c

and let f be the following PS-expression—in fact, an S-expression:

$$f(R) = \sigma_{(A=a)\wedge(B=b)\vee(A=a')\wedge(B\neq b)}(R) \qquad (a \neq a').$$

We claim that there is no V-table U PS-representing $f(\operatorname{Rep}(T))$. Indeed, let

$$g(R) = \pi_C(\sigma_{(A=a)\vee(A=a')}(R)).$$

Notice that $f(\operatorname{Rep}(T))$ is coinitial with

$$\mathscr{X} = \{\boxed{abc}\} \cup \{\boxed{a'dc} : d \in D(B)\backslash\{b\}\},$$

so that $f(\operatorname{Rep}(T))^g = \mathscr{X}^g = \{c\}$.

If there is a V-tuple $pqv \in U$, where $p \in D(A)$, then $p \in \operatorname{Rep}(U)^{\pi_A} \neq \mathscr{X}^{\pi_A} = \varnothing$. If for every $pqr \in U$, p is a variable (in particular, if $U = \varnothing$), then clearly $\operatorname{Rep}(U)^g = \varnothing$, although we have already noted that $f(\operatorname{Rep}(T))^g = \{c\}$. Hence, in both cases, U does not PS-represent $f(\operatorname{Rep}(T))$. □

The apparent contradiction in the inability of V-multitables to support PS-expressions can be intuitively explained in the following way: $\langle \mathscr{T}, \operatorname{Rep}, \Omega\rangle$ is a representation system if the class $\mathscr{T}$ of multitables is a "fixpoint" with respect to Ω-expressions, in the sense that if we take any Ω-expression **f** and any $\mathbf{T} \in \mathscr{T}$, then to Ω-represent **f**(Rep(**T**)) we do not need any more "representation power" than that available in $\mathscr{T}$. If we consider an arbitrary $\mathscr{T}' \supseteq \mathscr{T}$, there is no reason for $\mathscr{T}'$ to be also such a fixpoint.

7. *Conditional Tables*

By a conditional table we mean a V-table extended by one additional special column, called *con,* which contains for any tuple a *condition* from a set $\mathscr{C}$.

More precisely, $\mathscr{C}$ is the set of all expressions built up from atomic conditions of the form $(x = a)$, $(x = y)$, **false** and **true**, where for some $A \in \mathscr{U}$, $a \in D(A)$, $x, y \in V(A)$, by means of $\neg$, $\vee$, and $\wedge$. A condition is *positive,* if it does not contain $\neg$.

By a C-tuple on X we mean any mapping t defined on $X \cup \{\text{con}\}$ such that $t[X]$ is a V-tuple and $t(\text{con}) \in \mathscr{C}$. A *conditional table* (or *C-table*) on X is any finite set T of C-tuples on X. (Note: We assume that con is not part of $\alpha(t)$ or $\alpha(T)$.) A *conditional multitable* (or *C-multitable*) of type $\langle X_1, \ldots, X_n\rangle$ is any sequence $\mathbf{T} = \langle T_1, \ldots, T_n\rangle$ where T_i is a C-table on X_i, $1 \leq i \leq n$. The set of all C-multitables is denoted by $\mathscr{T}_C$.

Any valuation can be extended in the natural way to conditions.

For any valuation v, any C-table T on X, and any C-multitable $\mathbf{T} = \langle T_1, \ldots, T_n\rangle$ of type $\langle X_1, \ldots, X_n\rangle$, we define

$$v(T) = \{v(t[X]) : v(t(\text{con})) = \textbf{true}\}, \tag{7.1}$$
$$v(\mathbf{T}) = \langle v(T_1), \ldots, v(T_n)\rangle, \tag{7.2}$$

and finally

$$\operatorname{Rep}(\mathbf{T}) = \{\mathbf{r} \in \mathscr{R}(X_1, \ldots, X_n) : \text{there exists a valuation } v \text{ such that } v(\mathbf{T}) \subseteq \mathbf{r}\}. \tag{7.3}$$

We say that conditions $\gamma, \delta \in \mathscr{C}$ are *equivalent*, and we write $\gamma \approx \delta$, if $v(\gamma) = v(\delta)$ for any valuation v. It can easily be shown that $\gamma \approx \delta$ iff γ can be transformed into δ by using the axioms of Boolean algebra and the axioms of equality. It should be clear that if we (a) replace each of the conditions in a C-table T by an equivalent one, (b) delete all $t \in T$ with $t(\mathrm{con}) \approx$ false, and (c) replace some $t_1, \ldots, t_k \in T$ such that $t_1[X] = \cdots = t_k[X]$ $(X = \alpha(T))$ by a single C-tuple t such that $t[X] = t_1[X]$ and $t(\mathrm{con}) = \bigvee_{i=1}^{k} t_i(\mathrm{con})$, then the resulting C-table U will be Rep-equivalent to T (i.e., $\mathrm{Rep}(T) = \mathrm{Rep}(U)$, see Section 3). We freely make use of the equivalent transformations of this kind; in particular, we use rule (c) to *normalize* a C-table T on X, that is, to replace T by a Rep-equivalent C-table T^0 on X not containing different C-tuples agreeing on X. In what follows we assume all C-tables to be normalized.

THEOREM 7.1. *It is possible to correctly evaluate PSUJ-expressions over C-tables; more formally, $\langle \mathscr{T}_C, \mathrm{Rep}, \mathrm{PSUJ}\rangle$ is a representation system.*

PROOF. We give definitions of the relational operators over C-tables that inductively define $\mathbf{f}(\mathbf{T})$ for any C-multitable $\mathbf{T}$ and PSUJ-expression $\mathbf{f}(\alpha(\mathbf{T}) = \alpha(\mathbf{f}))$, in such a way that

$$\mathbf{f}(\mathrm{Rep}(\mathbf{T})) \equiv_{\mathrm{PSUJ}} \mathrm{Rep}(\mathbf{f}(\mathbf{T})). \tag{7.4}$$

For any C-multitable $\mathbf{T} = \langle T_1, \ldots, T_n\rangle$ we put

$$\mathrm{pr}_i(\mathbf{T}) = T_i, \qquad 1 \le i \le n, \tag{7.5}$$

and for any C-tables T, W on X, Z, respectively, we define

$$\pi_Y(T) = \{t[Y \cup \{\mathrm{con}\}] : t \in T\}^0, \tag{7.6}$$
$$\sigma_E(T) = \{\sigma_E(t) : t \in T\}, \tag{7.7}$$

where $\sigma_E(t)$ is the C-tuple on X with

$$\sigma_E(t)[X] = t[X],$$
$$\sigma_E(t)(\mathrm{con}) = t(\mathrm{con}) \wedge E(t).$$

($E(t)$ is the result of substituting $t(A)$ for A in E, for every $A \in X$), and

$$T \bowtie W = \{t \bowtie w : t \in T \wedge w \in W\}^0, \tag{7.8}$$

where $t \bowtie w$ is the C-tuple on $X \cup Z$ with

$$(t \bowtie w)(A) = \begin{cases} t(A) & \text{if } A \in X, \\ w(A) & \text{if } A \in Z \backslash X, \end{cases}$$
$$(t \bowtie w)(\mathrm{con}) = t(\mathrm{con}) \wedge w(\mathrm{con}) \wedge \bigwedge_{A \in X \cap Z} (t(A) = w(A)).$$

If $X = Z$, then

$$T \cup W = (T \cup W)^0, \tag{7.9}$$

where $\cup$ on the left-hand side denotes the relational union operator that we define on C-tables, and $\cup$ on the right-hand side is the usual set-theoretical union.

Finally, if $\mathbf{f} = \langle f_1, \ldots, f_k\rangle$, then

$$\mathbf{f}(\mathbf{T}) = \langle f_1(\mathbf{T}), \ldots, f_k(\mathbf{T})\rangle. \tag{7.10}$$

We now prove that under the definition of $\mathbf{f}(\mathbf{T})$ given by (7.5)–(7.10),

$$v(\mathbf{f}(\mathbf{T})) = \mathbf{f}(v(\mathbf{T})) \tag{7.11}$$

for any valuation v. (On the right-hand side, $\mathbf{f}(v(\mathbf{T}))$ is understood as the result of performing $\mathbf{f}$ over the multirelation $v(\mathbf{T})$ in the usual way.) Clearly, (7.11) implies $\mathrm{Rep}(\mathbf{f}(\mathbf{T})) \approx \mathbf{f}(\mathrm{Rep}(\mathbf{T}))$, which by Lemma 4.2, proves (7.4).

We show (7.11) by verifying that

$$\begin{aligned}
\nu(\mathrm{pr}_t(\mathbf{T})) &= \mathrm{pr}_t(\nu(\mathbf{T})),\\
\nu(\pi_Y(T)) &= \pi_Y(\nu(T)),\\
\nu(T \bowtie W) &= \nu(T) \bowtie \nu(W),\\
\nu(T \cup W) &= \nu(T) \cup \nu(W) \qquad (\alpha(T) = \alpha(W)),
\end{aligned}$$

for any C-multitable **T** and any C-tables T, W.

Let $\alpha(T) = X$, $\alpha(W) = Z$. In the case of projection, $t \in \nu(\pi_Y(T))$ iff there is a C-tuple $t' \in \pi_Y(T)$ such that $\nu(t'[Y]) = t$, $\nu(t'(\mathrm{con})) = \mathbf{true}$; that is, iff there exists a C-tuple $t'' \in T$ such that $\pi_Y(\nu(t''[X])) = t$, $\nu(t''(\mathrm{con})) = \mathbf{true}$, which is exactly the condition for t to be in $\pi_Y(\nu(T))$.

In the case of selection, $t \in \nu(\sigma_E(T))$ iff there is a $t' \in T$ such that $\nu(t'(\mathrm{con})) = \mathbf{true}$, $\nu(E(t')) = \mathbf{true}$, and $t = \nu(t'[X])$, that is, iff $t \in \sigma_E(\nu(T))$.

In the case of join, a tuple t of type $X \cup Z$ is in $\nu(T \bowtie W)$ iff there are tuples $t' \in T$, $w' \in W$ such that $\nu((t' \bowtie w')(\mathrm{con})) = \mathbf{true}$, $\nu((t' \bowtie w'([X \cup Z]) = t$, and this is clearly equivalent to $t \in \nu(T) \bowtie \nu(W)$.

The case of union is obvious. □

Example 7.1. Let us evaluate

$$f(R) = \sigma_{C=c}(\pi_{AC}(\pi_{AB}(R) \bowtie \pi_{BC}(R)))$$

over the following C-table·

$T =$

A	B	C	con
a	b	z	$z \neq c$
a	y	c	$y \neq b$
x	b	c	$x \neq a$

We have

$U = \pi_{AB}(T) \bowtie \pi_{BC}(T) =$

A	B	C	con
a	b	z	$(z \neq c) \wedge (z \neq c) \wedge (b = b)$
a	b	c	$(z \neq c) \wedge (y \neq b) \wedge (b = y)$
a	b	c	$(z \neq c) \wedge (x \neq a) \wedge (b = b)$
a	y	z	$(y \neq b) \wedge (z \neq c) \wedge (y = b)$
a	y	c	$(y \neq b) \wedge (y \neq b) \wedge (y = y)$
a	y	c	$(y \neq b) \wedge (x \neq a) \wedge (y = b)$
x	b	z	$(x \neq a) \wedge (z \neq c) \wedge (b = b)$
x	b	c	$(x \neq a) \wedge (y \neq b) \wedge (b = y)$
x	b	c	$(x \neq a) \wedge (x \neq a) \wedge (b = b)$

$=$

A	B	C	con
a	b	z	$(z \neq c)$
a	b	c	$(z \neq c) \wedge (y \neq b)$
a	y	c	$(y \neq b)$
x	b	z	$(x \neq a) \wedge (z \neq c)$
x	b	c	$(x \neq a)$

$W = \pi_{AC}(U) =$

A	C	con
a	z	$(z \neq c)$
a	c	$(y \neq b)$
x	z	$(x \neq a) \wedge (z \neq c)$
x	c	$(x \neq a)$

and finally

$\sigma_{C=c}(W) =$

A	C	con
a	z	$(z \neq c) \wedge (z = c)$
a	c	$(y \neq b) \wedge (c = c)$
x	z	$(x \neq a) \wedge (z \neq c) \wedge (z = c)$
x	c	$(x \neq a) \wedge (c = c)$

$\equiv$

A	C	con
a	c	$(y \neq b)$
x	c	$(x \neq a)$

□

Let T be a C-table on X. As usual we assume that no two different tuples in T agree on X. We define the *unconditional part* of T to be

$$T_* = \{t \in T : t(\text{con}) \approx \mathbf{true}\}.$$

The unconditional part of a C-multitable $\mathbf{T} = \langle T_1, \ldots, T_n \rangle$ is defined as $\mathbf{T}_* = \langle T_{1*}, \ldots, T_{n*} \rangle$.

A C-multitable $\mathbf{T} = \langle T_1, \ldots, T_n \rangle$ is called *positive* if for every T_i and every $t \in T_i$, $t(\text{con})$ is positive.

LEMMA 7.1. *Let all attribute domains be infinite, and let* $\mathbf{T}$ *be a positive C-multitable. Then*

$$\bigcap \mathit{Rep}(\mathbf{T}) = \bigcap \mathit{Rep}(\mathbf{T}_*).$$

PROOF. Obviously, we may restrict ourselves to the case where $\mathbf{T}$ is a single C-table, T.

Clearly $\mathrm{Rep}(T) \subseteq \mathrm{Rep}(T_*)$, and consequently $\bigcap \mathrm{Rep}(T_*) \subseteq \bigcap \mathrm{Rep}(T)$, so that it is sufficient to prove the converse inclusion.

Suppose $t \notin \bigcap \mathrm{Rep}(T_*)$. It means that there is a valuation v such that $t \notin v(T_*)$. Let v' be a valuation such that $v'(x) = v'(y)$ for $x \neq y$ and for every variable x appearing in T, $v'(x)$ does not appear in T or t (the existence of such a valuation v' is an obvious consequence of the assumption that the attribute domains are infinite). Clearly, v' makes all atomic conditions (which are positive, by our assumption) false, and consequently it makes false all conditions $t(\text{con})$, $t \in T \backslash T_*$. Since $t \notin v(T_*)$, it follows that T_* does not contain any tuple agreeing with t on $\alpha(t)$. The valuation v' associates constants different from those appearing in t to variables in T, so that it cannot produce t; that is, $t \notin v'(T)$. Consequently, $t \notin \bigcap \mathrm{Rep}(T)$. □

We are now going to prove, as we promised in the previous section, that $\langle \mathcal{T}_V, \mathrm{Rep}, \mathrm{PS^+UJ} \rangle$ is a representation system.

A C-multitable $\mathbf{T}$ is called *unconditional* if $\mathbf{T}_* = \mathbf{T}$. Let $\mathbf{T} = \langle T_1, \ldots, T_n \rangle$ be an unconditional C-multitable and let $\mathbf{U}$ be the V-multitable obtained from $\mathbf{T}$ by deleting the con column from every T_i, $1 \le i \le n$. Then

$$\mathrm{Rep}(\mathbf{T}) = \mathrm{Rep}(\mathbf{U}),$$

where Rep(**T**) is computed according to (7.1)–(7.3) and Rep(**U**) according to (6.1)–(6.4). In what follows we always identify an unconditional C-multitable with the corresponding V-multitable. To avoid confusion, we write $\mathbf{f}^C(\mathbf{T})$ and $\mathbf{f}^V(\mathbf{T})$ for an expression **f** evaluated for an unconditional C-multitable **T** (identified with a V-multitable) according to the rules given for C-tables (see (7.5)–(7.10)) and V-tables (see (6.5)–(6.9)), respectively.

LEMMA 7.2. *Let all attribute domains be infinite, let* **T** *be a positive C-multitable and let* **f** *be a PS$^+$UJ-expression with* $\alpha(\mathbf{f}) = \alpha(\mathbf{T})$. *Then*

$$\mathbf{f}^V(\mathbf{T}_*) = \mathbf{f}^C(\mathbf{T})_*.$$

PROOF. It is sufficient to prove that for any positive C-multitable $\mathbf{T} = \langle T_1, \ldots, T_n\rangle$

$$\mathrm{pr}_i^V(\mathbf{T}_*) = \mathrm{pr}_i^C(\mathbf{T})_*, \qquad 1 \le i \le n \tag{7.12}$$

and that for any positive C-tables T, W

$$\begin{aligned} \pi_Y^V(T_*) &= \pi_Y^C(T)_*, & & (7.13)\\ \sigma_E^V(T_*) &= \sigma_E^C(T)_* \qquad \text{for} \quad E \text{ positive}, & & (7.14)\\ T_* \bowtie^V W_* &= (T \bowtie^C W)_*, & & (7.15)\\ T_* \cup^V W_* &= (T \cup^C W)_*, & & (7.16) \end{aligned}$$

$(X = \alpha(T), Z = \alpha(W))$.

Before we verify these equalities, let us note the following simple fact concerning positive conditions:

For any positive conditions γ, δ

$$\gamma \vee \delta \approx \mathbf{true} \Rightarrow \gamma \approx \mathbf{true} \qquad \text{or} \qquad \delta \approx \mathbf{true}. \tag{7.17}$$

Indeed, it is easy to see that, if $\gamma \not\approx$ **true** and $\delta \not\approx$ **true**, then $v(\gamma \vee \delta) =$ **false**, where v is any valuation such that $v(x) \ne v(y)$ for $x \ne y$ and $v(x)$ does not appear in $\gamma \vee \delta$ for any $x \in V$.

Since (7.12) is obvious, we now consider the case of projection. A V-tuple t is in $\pi_Y^V(T_*)$ iff there is a $t' \in T_*$ such that $t'[Y] = t[Y]$, that is, if there is a $t'' \in T$ such that $t''[Y] = t[Y]$ and $t''(\mathrm{con}) \approx$ **true**. On the other hand, $t \in \pi_Y^C(T)_*$ iff there are tuples $t_1, \ldots, t_k \in T$, $k \ge 1$ such that $t_i[Y] = t[Y]$, $1 \le i \le k$ and $\bigvee_{i=1}^k t_i(\mathrm{con}) \approx$ **true**. But by (7.17) the last condition implies that $t_i(\mathrm{con}) \approx$ **true** for some i, and we obtain exactly the same condition as before.

In the case of selection, $\sigma_E^V(T_*)$ consists of all tuples $t \in T$ such that for every valuation v, $v(t(\mathrm{con})) =$ **true** and $v(E(t)) =$ **true**, which is exactly the condition for t to be in $\sigma_E^C(T)_*$. Notice that (7.14) holds true for any, not necessarily positive, condition E. The requirement that E be positive is needed only to guarantee that $\sigma_E^C(T)_*$ is positive, so that we could inductively prove the lemma using (7.12)–(7.16).

We now consider the case of join. It is obvious that $T_* \bowtie^V W_* \subseteq (T \bowtie^C W)_*$, so it is sufficient to prove the converse inclusion. Suppose $t \in (T \bowtie^C W)_*$. Then there are C-tuples $t_1, \ldots, t_k \in T$, $w_1, \ldots, w_k \in W$, $k \ge 1$, such that $t_i[X] = t[X]$,

$w_i[Z \backslash X] = t[Z \backslash X]$, $1 \le i \le k$, and $\bigvee_{i=1}^{k}(t_i \bowtie w_i)(\text{con}) \approx \textbf{true}$. By (7.17), for some i

$$t_i(\text{con}) \wedge w_i(\text{con}) \wedge \bigwedge_{A \in X \cap Z} (t_i(A) = w_i(A)) \approx \textbf{true}.$$

This obviously means that $t_i \in T_*$, $w_i \in W_*$, and $t_i[X \cap Z] = w_i[X \cap Z]$; that is, $t \in T_* \bowtie^V W_*$.

Finally, in the case of union, $T_* \cup^V W_* \subseteq (T \cup^C W)_*$, and $t \in (T \cup^C W)_*$ implies that either $t \in T_*$ or $t \in W_*$ or there are $t' \in T$, $w' \in W$ with $t'[X] = w'[X] = t[X]$, $t'(\text{con}) \vee w'(\text{con}) \approx \textbf{true}$. The last case is reduced, however, by (7.17), to either the first or the second one. □

We are now ready to give the promised proof of the fact that $\langle \mathcal{T}_V, \text{Rep}, \text{PS}^+\text{UJ} \rangle$ is a representation system.

PROOF OF THEOREM 6.1. We have to show that for any V-multitable $\mathbf{T}$ and any PS⁺UJ-expression $\mathbf{f}$ with $\alpha(\mathbf{f}) = \alpha(\mathbf{T})$,

$$\text{Rep}(\mathbf{f}^V(\mathbf{T})) \equiv_{\text{PS}^+\text{UJ}} \mathbf{f}(\text{Rep}(\mathbf{T}));$$

that is, for any PS⁺UJ-expression $\mathbf{g}$ with appropriate $\alpha(\mathbf{g})$,

$$\text{Rep}(\mathbf{f}^V(\mathbf{T}))^{\mathbf{g}} = \mathbf{f}(\text{Rep}(\mathbf{T}))^{\mathbf{g}}. \tag{7.18}$$

The left-hand side of (7.18) can be transformed as follows:

$$\text{Rep}(\mathbf{f}^V(\mathbf{T}))^{\mathbf{g}} = \bigcap_{\mathbf{r} \in \text{Rep}(\mathbf{f}^V(\mathbf{T}))} \mathbf{g}(\mathbf{r}) = \bigcap \mathbf{g}(\text{Rep}(\mathbf{f}^V(\mathbf{T}))).$$

By Theorem 7.1,

$$\mathbf{g}(\text{Rep}(\mathbf{f}^V(\mathbf{T}))) \equiv_{\text{PSUJ}} \text{Rep}(\mathbf{g}^C(\mathbf{f}^V(\mathbf{T}))),$$

which implies the equality of the intersections

$$\bigcap \mathbf{g}(\text{Rep}(\mathbf{f}^V(\mathbf{T}))) = \bigcap \text{Rep}(\mathbf{g}^C(\mathbf{f}^V(\mathbf{T}))).$$

By applying Lemma 7.1 and then Lemma 7.2 to the right-hand side we get

$$\begin{aligned}\bigcap \mathbf{g}(\text{Rep}(\mathbf{f}^V(\mathbf{T}))) &= \bigcap \text{Rep}(\mathbf{g}^C(\mathbf{f}^V(\mathbf{T}))_*) \\ &= \bigcap \text{Rep}(\mathbf{g}\mathbf{f}^V(\mathbf{T})).\end{aligned}$$

Again using Lemmas 7.2 and 7.1 and Theorem 7.1 we obtain

$$\begin{aligned}\bigcap \text{Rep}(\mathbf{g}\mathbf{f}^V(\mathbf{T})) &= \bigcap \text{Rep}(\mathbf{g}\mathbf{f}^C(\mathbf{T})_*) \\ &= \bigcap \text{Rep}(\mathbf{g}\mathbf{f}^C(\mathbf{T})) \\ &= \bigcap \mathbf{g}\mathbf{f}(\text{Rep}(\mathbf{T}))\end{aligned}$$

which is the same as the right-hand side of (7.18):

$$\mathbf{f}(\text{Rep}(\mathbf{T}))^{\mathbf{g}} = \bigcap_{\mathbf{r} \in \mathbf{f}(\text{Rep}(\mathbf{T}))} \mathbf{g}(\mathbf{r}) = \bigcap \mathbf{g}\mathbf{f}(\text{Rep}(\mathbf{T})). \qquad \square$$

To conclude this section, we note that

$$\mathbf{f}(v(\mathbf{T})) \supseteq v(f(\mathbf{T})) \tag{7.19}$$

for any PS⁺UJ-expression $\mathbf{f}$, any V-multitable $\mathbf{T}$ ($\alpha(\mathbf{T}) = \alpha(\mathbf{f})$), and any valuation v. This is proved easily by induction on the complexity of $\mathbf{f}$. An immediate corollary is

$$\text{Rep}(\mathbf{f}(\mathbf{T})) \supseteq \mathbf{f}(\text{Rep}(\mathbf{T})). \tag{7.20}$$

An important fact about (7.20) is that it does not require any assumptions on the cardinality of attribute domains. Notice that, in the terminology of the Introduction, (7.20) says that the representation system based on V-multitables is always safe: Rep(f(**T**)) includes all "possible" f(r*), r* ∈ Rep(**T**). The assumption that the attribute domains are infinite is only needed to prove its completeness.

8. *Renaming of Attributes, Conjunctive Queries*

It may be noted that the relational algebra based on Ω = PSUJD and a fixed finite set $\mathscr{U}$ of attributes is not relationally complete in the sense of Codd [6]. This is because our version of relational algebra is equivalent in expressive power to a many-sorted predicate calculus with only one variable available for each sort, and certain formulas of the usual (many-sorted) predicate calculus inherently need many "auxiliary variables" (cf. [14]). One way of making our algebra relationally complete is to add the operation of renaming attributes, and to consider an infinite set of attributes $\mathscr{U}$, which contains an infinite number of attributes for each sort (i.e., for every $A \in \mathscr{U}$, there exists different $A_1, A_2, \ldots$ with $D(A_i) = D(A)$). It is obvious that

$$s_B^A(\mathrm{Rep}(T)) = \mathrm{Rep}(s_B^A(T)), \tag{8.1}$$

where $s_B^A(T)$ (with T either a Codd table, V-table, or C-table) is defined in exactly the same way as $s_B^A(r)$. By (8.1), most of our results can easily be extended by adding renaming (R) to the set Ω of relational operators. In particular, by Theorem 6.1, we have the following corollary.

COROLLARY 8.1. *$\langle \mathscr{T}_V$, Rep, PS$^+$UJR$\rangle$ is a representation system.*

We now discuss another version of the relational algebra (see, e.g., [29]) where the type of a relation is defined as a *sequence*, rather than set, of attributes (repetitions are allowed). The relational operators are modified in the following way. In the projection operator, instead of a target attribute set we have a target sequence of positions (i.e., numbers of columns of the argument relation); in the selection condition, we refer to positions rather than to attributes; and instead of the natural join we consider the Cartesian product. The Cartesian product of two relations, r, s, of types $\langle A_1, \ldots, A_n\rangle$ and $\langle B_1, \ldots, B_m\rangle$, respectively, is a relation of type $\langle A_1, \ldots, A_n, B_1, \ldots, B_m\rangle$ defined as

$$r \times s = \{\langle a_1, \ldots, a_n, b_1, \ldots, b_m\rangle : \langle a_1, \ldots, a_n\rangle \in r \wedge \langle b_1, \ldots, b_m\rangle \in s\}. \tag{8.2}$$

Note that any relation, in the modified sense, of type $\langle A_1, \ldots, A_n\rangle$ can be uniquely represented by the usual relation of type $\{\langle A_1, 1\rangle, \ldots, \langle A_n, n\rangle\}$, and that (8.2) can be expressed by first renaming the attributes of s so that it is of type $\{\langle B_1, n+1\rangle, \ldots, \langle B_m, n+m\rangle\}$ and then performing the natural join. By Corollary 8.1, this easily implies the following theorem (where X stands for the Cartesian product):

THEOREM 8.1. *$\langle \mathscr{T}_V$, Rep, PS$^+$UX$\rangle$ is a representation system.*

Conversely, the join can easily be expressed by the Cartesian product, restriction, and projection, so that, by Theorem 4.7, the Codd tables cannot support Ω = PXE (E stands for restriction).

THEOREM 8.2. *$\langle \mathscr{T}_{@}$, Rep, PXE$\rangle$ is not a representation system.*

(Clearly, in Theorems 8.1 and 8.2 the notions of a V-table and Codd table are modified into a "column-ordered" form, in the same way as it was done for relations.)

Another important fact implied by Corollary 8.1 is that the system based on V-tables can handle arbitrary conjunctive queries [4]. Roughly speaking, a conjunctive query is defined by a conjunction of atomic relational formulas of the form $R_i(x_1, \ldots, x_p)$ (the R_i's are predicates corresponding to database relations), where each of the x_i is either a constant or variable, preceded by existential quantifiers binding some of the variables appearing in the conjunction. The mapping defined by a conjunctive query q can be defined, for any database state $\mathbf{r}$, by

$$q(\mathbf{r}) = \{v(s) : v \text{ is a valuation such that } v(\mathbf{Q}) \subseteq \mathbf{r}\} \tag{8.3}$$

where

$\mathbf{Q}$ is a suitable V-multitable (V-tuples appearing in $\mathbf{Q}$ correspond in the natural way to the atomic relational formulas in q; $\alpha(\mathbf{Q}) = \alpha(\mathbf{r}) = \alpha(q)$),
s is a V-tuple such that all variables occurring in s occur in $\mathbf{Q}$ ($\alpha(s) = \beta(q)$).

LEMMA 8.1. *For any conjunctive query q there is a PS^+JR-expression f such that $q(\mathbf{r}) = f(\mathbf{r})$ for every $\mathbf{r}$ ($\alpha(\mathbf{r}) = \alpha(f) = \alpha(q)$).*

SKETCH OF PROOF. Call a variable *repeated* if it occurs in at least two V-tuples in $\mathbf{Q}$, s, and a *target* variable if it occurs in s (see (8.3)). For every repeated variable x, let A_x be an attribute not appearing in $\mathbf{Q}$. Let $\mathbf{Q} = \langle Q_1, \ldots, Q_n \rangle$. For any V-tuple u in Q_i, let f_u be an expression consisting of a relation symbol R_i preceded by (from right to left)

(i) selections $\sigma_{A=a}$, for every $A \in \alpha(u)$ with $u(A) = a$, $a \in D$,
(ii) restrictions $\sigma_{A=B}$, for all $A, B \in \alpha(u)$ with $u(A) = u(B) = x$, a variable,
(iii) projection π_Y, where Y contains exactly one attribute A for any repeated variable x appearing in u ($u(A) = x$),
(iv) renaming $s^A_{A_x}$, for every $A \in Y$, where $x = u(A)$.

Let $h = \pi_X(\bowtie_u f_u)$, where u ranges over all V-tuples in $\mathbf{Q}$ and $X = \{A_x : x \text{ is a target variable}\}$. The required expression f is obtained from h by suitably renaming the attributes in X. Some easy modifications are required if s contains constants or multiple occurrences of the same variable. (A similarity of this construction to "shallow expressions" of [32] may be noted.) □

By Lemma 8.1, we obtain the following corollary.

COROLLARY 8.2. *The representation system $\langle \mathcal{T}_V, Rep, PS^+UJR \rangle$ can support arbitrary conjunctive queries; that is, for any conjunctive query q and any V-multitable $\mathbf{T}$ with $\alpha(\mathbf{T}) = \alpha(q)$ there is a V-table U such that $Rep(U) \equiv_{PS^+UJR} q(Rep(\mathbf{T}))$.*

Here we denote $q(\text{Rep}(\mathbf{T})) = \{q(\mathbf{r}) : \mathbf{r} \in \text{Rep}(\mathbf{T})\}$. The V-table U is obtained by finding the PS$^+$JR-expression f equivalent to q (see Lemma 8.1) and putting $U = f(\mathbf{T})$. The V-table U will also be denoted by $q(\mathbf{T})$.

By using the fact that in computing $f(\mathbf{T})$ we treat variables and constants in exactly the same way, and that the equivalence between q and f does not depend on the nature of the attribute domains, one can easily prove the following fact generalizing (8.3):

THEOREM 8.3. *For any conjunctive query q, given by a V-multitable* $\mathbf{Q}$ *and V-tuple s, and for any V-multitable* $\mathbf{T}$ *with* $\alpha(\mathbf{T}) = \alpha(\mathbf{Q})$

$$q(\mathbf{T}) = \{\rho(s) : \rho \text{ is a generalized valuation such that } \rho(\mathbf{Q}) \subseteq \mathbf{T}\},$$

where by a generalized valuation we mean any mapping ρ: $V \to V \cup D$ such that $x \in V(A)$ implies $\rho(x) \in V(A) \cup D(A)$.

9. *Closed World Interpretation of Tables*

The interpretation of multitables considered so far has been based on the open-world assumption [26, 27], which was embodied in the property

$$\mathrm{r} \in \mathrm{Rep}(\mathbf{T}) \wedge \mathrm{s} \supseteq \mathrm{r} \Rightarrow \mathrm{s} \in \mathrm{Rep}(\mathbf{T})$$

(see (3.3)). We now consider a different interpretation, which we call the *closed world interpretation.*

Let $\mathbf{T}$ be a V-multitable. The closed world interpretation of $\mathbf{T}$ is defined as follows:

$$\mathrm{rep}(\mathbf{T}) = \{\mathrm{r} : \text{there exists a valuation } \nu \text{ such that } \nu(\mathbf{T}) = \mathrm{r}\}, \qquad (9.1)$$

where $\nu(\mathbf{T})$ is defined as before (see (6.1)–(6.3)). Similarly, for any C-multitable $\mathbf{T}$ we define rep($\mathbf{T}$) by formula (9.1), with $\nu(\mathbf{T})$ defined by (7.1) and (7.2). In the case of a Codd multitable $\mathbf{T}$, we define

$$\mathrm{rep}(\mathbf{T}) = \mathrm{rep}(\mathbf{U}),$$

where $\mathbf{U}$ is any V-multitable obtained by replacing every occurrence of @ in $\mathbf{T}$ by a different variable. It seems that what Codd meant in [6] was in fact the closed world interpretation of Codd tables.

Notice that if a tuple t cannot be obtained by replacing nulls by constants in any @-tuple of a Codd table $\mathbf{T}$, then $\mathbf{T}$ represents the "negative information" that the relationship expressed by t definitely does not hold, a fact not representable under the open world interpretation. The situation is similar in the case of V-tables and C-tables.

Clearly, for all three kinds of multitables,

$$\mathrm{rep}(\mathbf{T}) \subseteq \mathrm{Rep}(\mathbf{T});$$

moreover, rep($\mathbf{T}$) and Rep($\mathbf{T}$) are coinitial.

Let $\Omega \subseteq \mathrm{PSS^{+}UJR}$, $\mathbf{f}$ be an Ω-expression, $\mathbf{T}$ be a multitable of type $\alpha(\mathbf{f})$, and $\mathbf{U}$ be a multitable of type $\beta(\mathbf{f})$. Since rep($\mathbf{U}$) $\approx$ Rep($\mathbf{U}$), rep($\mathbf{T}$) $\approx$ Rep($\mathbf{T}$), by Lemma 4.2 we obtain

$$\mathrm{rep}(\mathbf{U}) \equiv_{\Omega} \mathrm{Rep}(\mathbf{U})$$

and

$$\mathbf{f}(\mathrm{rep}(\mathbf{T})) \equiv_{\Omega} \mathbf{f}(\mathrm{Rep}(\mathbf{T})).$$

Hence,

$$\mathrm{rep}(\mathbf{U}) \equiv_{\Omega} \mathbf{f}(\mathrm{rep}(\mathbf{T})) \quad \text{iff} \quad \mathrm{Rep}(\mathbf{U}) \equiv_{\Omega} \mathbf{f}(\mathrm{Rep}(\mathbf{T})).$$

This equivalence immediately implies that all results concerning representation systems developed under the open-world assumption carry over to the closed-world interpretation:

THEOREM 9.1. *Let $\Omega \subseteq PSS^{+}UJR$ and let $\mathcal{T}$ be either $\mathcal{T}_{@}$ or $\mathcal{T}_V$ or $\mathcal{T}_C$. Then*

$$\begin{aligned} &\langle \mathcal{T}, Rep, \Omega\rangle \text{ is a representation system} \\ &\textit{iff} \quad \langle \mathcal{T}, rep, \Omega\rangle \text{ is a representation system.}\end{aligned}$$

Moreover, the correct operations on tables are the same under both interpretations.

It turns out that in the case of conditional tables we can handle all relational operations, including the difference, thus supporting the full strength of the relational algebra. Indeed, let T, U be two C-tables, $\alpha(T) = \alpha(U) = X$. For any $t \in T$, let us define t_U to be a C-tuple, $\alpha(t_U) = X$, such that

$$\begin{aligned} t_U[X] &= t[X], \\ t_U(\text{con}) &= t(\text{con}) \wedge \bigwedge_{u\in U} \bigvee_{A\in X} (t(A) \neq u(A)). \end{aligned}$$

If we define

$$T - U = \{t_U : t \in T\},$$

then it is easily seen that

$$v(T - U) = v(T) - v(U)$$

for any valuation v, which, combined with (7.11), implies that

$$v(\mathbf{f}(\mathbf{T})) = \mathbf{f}(v(\mathbf{T}))$$

for any PSUJRD-expression $\mathbf{f}$, any C-multitable $\mathbf{T}$ ($\alpha(\mathbf{T}) = \alpha(\mathbf{f})$), and any valuation v. Consequently,

$$\text{rep}(\mathbf{f}(\mathbf{T})) = \mathbf{f}(\text{rep}(\mathbf{T})), \tag{9.2}$$

that is, diagram (3.1) commutes. Let us state this as a theorem.

THEOREM 9.2. *$\langle \mathcal{T}_C$, rep, PSUJRD$\rangle$ is a representation system. Moreover, all relational operators can be defined on C-tables in such a way that* (9.2) *holds.*

10. *Conclusions*

We have proposed a general condition that should be satisfied if we want to evaluate relational expressions over "tables with nulls" in a semantically correct way.

Suppose that an instance of an incomplete information relational database is given by a multitable **T**, and let **f** be a relational query. While computing the response to **f**, we think of an unknown $\mathbf{r}^* \in \text{Rep}(\mathbf{T})$—corresponding to the true state of the real world—being transformed by **f**, and we require that **f**(**T**) contain enough information to determine all tuples that surely (i.e., no matter which element of Rep(**T**) the multirelation $\mathbf{r}^*$ is) appear in $\mathbf{f}(\mathbf{r}^*)$.

In this way our approach is similar in spirit to the *external interpretation* and *lower value* $\|\cdot\|_*$ in [20] (see also [22]).

Note that the tuples that surely appear in $\mathbf{f}(\mathbf{r}^*)$ are given by $\text{Rep}(\mathbf{T})^{\mathbf{f}}$, and it is easy to see that $\text{Rep}(\mathbf{T})^{\mathbf{f}}$ can be obtained by deleting from **f**(**T**) all tuples containing nulls, both in the case of Codd tables and V-tables.

Another possibility is to treat **f**(**T**), rather than $\text{Rep}(\mathbf{T})^{\mathbf{f}}$, as the response to query **f**. This has the advantage of providing more information; for instance, a tuple *a@c* (or *ayc*) appearing in the response can easily be shown to indicate that a tuple of the form *abc*, $b \in D(B)$ surely appears in $\mathbf{f}(\mathbf{r}^*)$. Moreover, this approach makes it possible to treat a response to query f as an intermediate step in computing the correct response to a more complex query gf (note that in general it is *not* possible to determine $\text{Rep}(\mathbf{T})^{\mathbf{gf}}$ on the basis of $\text{Rep}(\mathbf{T})^{\mathbf{f}}$ and g).

It should be noted that $\text{Rep}(\mathbf{T})^{\mathbf{f}} = \text{rep}(\mathbf{T})^{\mathbf{f}}$, so that by Theorem 9.1 the process of computing the response to a query is exactly the same under both open and closed world assumptions.

The results of the paper show that the devices suitable for representing incomplete information heavily depend on what processing of the information we are going to correctly perform, or, in more concrete terms, what relational operators are allowed. Our approach stresses the requirement for a correct evaluation of relational expressions, rather than just single relational operators, as was usually the case in previous attempts in the literature to correctly handle null values.

From the practical point of view an especially appealing system seems to be one based on V-tables, supporting projection, positive selection, union, join, and renaming, which allows for processing arbitrary conjunctive queries. The reason is that all these relational operators and queries can be evaluated over V-tables in *exactly the same way* as in the case of the usual relations.

There are several important problems that have not been treated here. One is how to handle dependencies in our framework. It turns out that functional, multivalued, and join dependencies—more exactly, arbitrary implicational dependencies [2] ("generalized dependencies" in [29])—can be *represented* in any V-table, in the sense that for any V-table T and any set Σ of such dependencies, there is a V-table U such that $\text{Rep}(U) \equiv_{\text{PS}^+\text{UJ}} \text{Rep}(T) \cap \text{Sat}(\Sigma)$, where $\text{Sat}(\Sigma)$ is the set of all relations satisfying the dependencies in Σ (see [12, 13, 16]). This is another argument for the usefulness of the representation system based on V-tables.

Another interesting topic is the relation between V-tables and the tableaux of Aho, Sagiv, and Ullman [1]. Roughly speaking, a tableau for a relational PSJ-expression f (with only selection of the form $\sigma_{A=a}$ allowed) can be treated as a V-table representing the inverse image of the summary of the tableau with respect to f (see [15]). It also turns out that for any PSJ-expression f (with only selection of the form $\sigma_{A=a}$) and every V-multitable $\mathbf{T}$ ($\alpha(\mathbf{T}) = \alpha(\mathbf{f})$), there is a V-multitable $\mathbf{U}$ such that $\text{Rep}(\mathbf{U}) = \mathbf{f}^{-1}(\text{Rep}(\mathbf{T}))$. This fact has several interesting applications (see [17]).

ACKNOWLEDGMENTS. Several comments and suggestions by J. Loś, W. M Turski, and the referees are gratefully acknowledged.

REFERENCES

1. AHO, A.V., SAGIV, Y., AND ULLMAN, J.D. Equivalences among relational expressions. *SIAM J Comput 8*, 2 (May 1979), 218–246.
2. BEERI, C., AND VARDI, M.Y. Formal systems for tuple and equality generating dependencies. *SIAM J Comput 13*, 1 (Feb. 1984), 76–98.
3. BISKUP, J. A formal approach to null values in database relations. In *Advances in Database Theory*, H. Gallaire, J. Minker, and J.M. Nicolas, Eds. Plenum Press, New York, 1981, pp. 299–341.
4. CHANDRA, A.K., AND MERLIN, P.M. Optimal implementation of conjunctive queries in relational data bases. In *Proceedings of the 9th Annual ACM Symposium on Theory of Computing* (Boulder, Col., May 2-4). ACM, New York, 1977, pp. 77–90.
5. CODD, E.F. A relational model for large shared data banks. *Commun ACM 13*, 6 (June 1970), 377–387.
6. CODD, E.F. Relational completeness of data base sublanguages. In *Data Base Systems*, R. Rustin, Ed. Prentice-Hall, Englewood Cliffs, N.J., 1972, pp. 65–98.

7. CODD, E. F. Understanding relations (Installment #7). *FDT Bull. of ACM-SIGMOD 7*, 3–4 (Dec. 1975), 23–28.
8. CODD E. F. Extending the database relational model to capture more meaning. *ACM Trans Database Syst 4*, 4 (Dec. 1979), 397–434.
9. GAREY, M. R., AND JOHNSON, D. S. *Computers and Intractability: A Guide to the Theory of NP-completeness*. Freeman, San Francisco, Calif., 1979.
10. GRANT, J. Null values in a relational data base. *Inf Process. Lett. 6*, 5 (Oct. 1977), 156–157.
11. HOROWITZ, E., AND SAHNI, S. *Fundamentals of Computer Algorithms* Computer Science Press, Potomac, Md., 1979.
12. IMIELIŃSKI, T. Problems of representing information in relational databases (in Polish). Ph.D. Thesis, Institute of Computer Science, Polish Academy of Sciences, 1981.
13. IMIELIŃSKI, T., AND LIPSKI, W. On representing incomplete information in a relational data base. In *Proceedings of the 7th International Conference on Very Large Data Bases* (Cannes, France, Sept. 9–11) ACM, New York, 1981, pp. 388–397.
14. IMIELIŃSKI, T., AND LIPSKI, W. The relational model of data and cylindric algebras. *J. Comput. System Sci. 28*, 1 (Feb. 1984), 80–102.
15. IMIELIŃSKI, T., AND LIPSKI, W. A technique for translating states between database schemata. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (Orlando, Fla., June 2–4). ACM, New York, 1982, pp. 61–68.
16. IMIELIŃSKI, T., AND LIPSKI, W. Incomplete information and dependencies in relational databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (San Jose, Calif., May 23–26). ACM, New York, 1983, pp. 178–184.
17. IMIELIŃSKI, T., AND LIPSKI, W. Inverting relational expressions—a uniform and natural technique for various database problems. In *Proceedings of the ACM SIGACT-SIGMOD Symposium on Principles of Database Systems* (Atlanta, Ga., March 21–23). ACM, New York, 1983, pp. 305–311.
18. LACROIX, M., AND PIROTTE, A. Generalized joins. *ACM-SIGMOD Record 8*, 3 (Sept. 1976), 14–15.
19. LIPSKI, W. Informational systems with incomplete information. In *Proceedings of the 3rd International Colloquium on Automata, Languages and Programming* (Edinburgh, Scotland, July 20–23). Edinburgh University Press, Edinburgh, Scotland, 1976, pp. 120–130.
20. LIPSKI, W. On semantic issues connected with incomplete information databases. *ACM Trans Database Syst 4*, 3 (Sept. 1979), 262–296.
21. LIPSKI, W. On databases with incomplete information. *J. ACM 28*, 1 (Jan. 1981), 41–70.
22. LIPSKI, W. Logical problems related to incomplete information in databases. Tech. Rep. 138, Laboratoire de Recherche en Informatique, Université de Paris-Sud, Centre d'Orsay, Sept. 1983.
23. LIPSKI, W. On relational algebra with marked nulls. In *Proceedings of the 3rd ACM SIGACT-SIGMOD Symposium on Principles of Database Systems* (Waterloo, Ont., Canada, April 2–4). ACM, New York, 1984, pp. 201–203.
24. MAIER, D., ULLMAN, J. D., AND VARDI, M. Y. On the foundations of the universal relation model. *ACM Trans. Database Syst. 9*, 2 (June 1984), 283–308.
25. REINGOLD, E. M., NIEVERGELT, J., AND DEO, N. *Combinational Algorithms: Theory and Practice.* Prentice Hall, Englewood Cliffs, N.J., 1977.
26. REITER, R. On closed world databases. In *Logic and Data Bases*, H. Gallaire and J. Minker, Eds. Plenum Press, New York, 1978, pp. 55–76.
27. REITER, R. Towards a logical reconstruction of relational database theory. In *Conceptual Modelling, Perspectives from Artificial Intelligence, Databases and Programming Languages*, M. L. Brodie, J. Mylopoulos, and J. Schmidt, Eds. Springer-Verlag, New York, 1984, pp. 191–233.
28. SIKLÓSSY, L. Efficient query evaluation in relational databases with missing values. *Inf. Process. Lett 13*, 4/5 (End 1981), 160–163.
29. ULLMAN, J. D. *Principles of Database Systems.* 2nd ed. Computer Science Press, Potomac, Md., 1982.
30. VASSILIOU, Y. Null values in data base management: A denotational semantics approach. In *Proceedings of the ACM-SIGMOD International Symposium on Management of Data* (Boston, Mass., May 30–June 1). ACM, New York, 1979, pp. 162–169.
31. VASSILIOU, Y. Functional dependencies and incomplete information. In *Proceedings of the 6th International Conference on Very Large Data Bases* (Montreal, Ont., Canada, Oct. 1–3). ACM, New York, 1980, pp. 260–269.
32. YANNAKAKIS, M., AND PAPADIMITRIOU, C. H. Algebraic dependencies. *J. Comput Syst. Sci. 25*, 1 (Aug. 1982), 2–41.

33. Zaniolo, C. Relational views in a data base system support for queries. In *Proceedings of the IEEE Computer Software and Applications Conference* (Chicago, Ill., Nov. 8–11). IEEE, New York, 1977, pp. 267–275.
34. Zaniolo, C. Database relations with null values. In *Proceedings of the ACM SIGACT-SIGMOD Symposium on Principles of Database Systems* (Los Angeles, Calif., March 29–31). ACM, New York, 1982, pp. 27–33.

RECEIVED APRIL 1982; REVISED MARCH 1984; ACCEPTED MARCH 1984