BIT 32 (1992), 413–429.

# INFERRING NULL JOIN DEPENDENCIES IN RELATIONAL DATABASES

MARK LEVENE and GEORGE LOIZOU

*Department of Computer Science,*
*University College London,*
*Gower Street, London,*
*WC1E 6BT, U.K.*

*Department of Computer Science,*
*Birkbeck College,*
*Malet Street, London,*
*WC1E 7HX, U.K.*

## Abstract.

The inference problem for data dependencies in relational databases is the problem of deciding whether a set of data dependencies logically implies another data dependency. For join dependencies (JDs), the inference problem has been extensively studied by utilising the well-known chase procedure. We generalise JDs to null join dependencies (NJDs) that hold in relations which may contain null values. In our model for incomplete information we allow only a single unmarked null value denoted by *null*. This allows us to solve the inference problem for NJDs by extending the chase procedure to the or-chase procedure. In order to define the or-chase procedure we generalise relations with nulls to or-relations which contain a limited form of disjunctive information. The main result of the paper shows that the inference problem for NJDs, including embedded NJDs (which are a special case of NJDs), is decidable; this is realised via the or-chase procedure.

*CR Classification:* H.2.1.

*Key words:* null values, null-relations, or-relations, null join dependencies, or-chase procedure.

## 1. Introduction.

We address the problem of making inferences by using integrity constraints that must be enforced in a database. Herein we assume a relational database, which may be incomplete, and deal with a subclass of integrity constraints called data dependencies [3, 6, 21]. Our model for incomplete information contains a single unmarked null value, *null*. This model of unmarked nulls is a special case of the more general model of incomplete information based on multiple marked nulls [9, 10], wherein null values are marked with a distinguishing index and are allowed to be equal if their indexes are equal. We generalise the class of *join dependencies* (JDs) [2, 19] so as

Received July 1990. Revised March 1992.

to hold in relations which may contain *null*; such relations are called *null-relations* and the JDs that hold therein are called *null join dependencies* (NJDs).

The *inference problem* for data dependencies is the problem of *deciding* whether a set of data dependencies logically implies another data dependency. An *inference procedure* for a class of data dependencies is said to be *sound* if all the inferences made by this procedure are always true, and is said to be *complete* if all possible inferences that are true are made by this procedure. Although no sound and complete set of inference rules has been shown for JDs, an inference procedure, called the *chase procedure* [1, 3, 6, 16, 20], has been shown to be sound for JDs and complete for *full JDs*, i.e. JDs whose context is the universal set of attributes, $U$. On the other hand, the chase procedure is not complete for *embedded JDs*, i.e. JDs whose context is a proper subset $W \subset U$, since, in general, their inference problem is known to be only partially decidable [3, 6, 17]. We extend the chase procedure to the *or-chase procedure* in order to solve the inference problem for NJDs. Before doing so, we generalise null-relations to *or-relations*, which allow a limited form of disjunctive information [7, 10, 15]. The or-chase procedure is then defined in order to make inferences from an or-relation with respect to a set of NJDs. The main result of the paper shows that the or-chase procedure is a sound and complete inference procedure for NJDs. Thus, in our formalism the inference problem for embedded NJDs (as well as full NJDs) is decidable. Our stronger result is due to the use of a single unmarked null (cf. [1, 20]) as opposed to the use of multiple marked nulls as was done in previous formalisms for embedded data dependencies [3, 6, 16, 19]. The *or-chase* procedure can also be utilised for testing the satisfaction of a set of NJDs in a null-relation. Finally, our formalism is more general than the one presented in [11], since we make no assumptions about the acyclicity or cyclicity of the database scheme.

The rest of the paper is organised as follows. In Section 2 we define our underlying model for incomplete information. In Section 3 we introduce NJDs and present their inference problem. In Section 4 we solve the inference problem for NJDs by defining the or-chase procedure which is applied to or-relations with respect to a set of NJDs. In Section 5 we give our concluding remarks.

## 2. The single unmarked null model for incomplete information.

In this section we introduce our model for incomplete information by including a single unmarked null value, *null*, in attribute domains, leading to the definition of *null-relations*. We then define a partial ordering on tuples in null-relations and generalise it to the *Hoare powerdomain ordering* [18] on null-relations. This allows us to formalise the notion of the relative information content of tuples and null-relations. In our semantics of null values we define the *inequality rule for nulls* which states that *null* $\neq$ *null*. Finally, we introduce a more general class of relations, termed *or-relations*, which allow a limited form of disjunctive information. The motivation

for defining or-relations is to obtain a compact representation of a set of null-relations. This is utilised in Section 4 in order to solve the inference problem for NJDs.

Let $D$ be a countable flat domain comprising atomic values (also referred to as *total values*) and a bottom element $\perp$. We define a partial order $\leq$ on $D$ as follows: $\forall v_i, v_j \in D$, $v_i \leq v_j$ if and only if $v_i = v_j$ or $v_i = \perp$ [18].

We interpret the bottom element as being the unmarked null value, *null*, in the sense that *null* ($\perp$) contains less information than any other value in $D$. Thus, in our formalism we consider only one null value, *null*, which will be a member of all the available domains. We observe that our formalism can accommodate the inclusion of several particular unmarked null types in attribute domains, such as: *value unknown* [4, 9], *value does not exist* [4, 12], and *no information* [22]. This could be done by considering a *semi-lattice domain* of null values [18].

Let $U = \{A_1, A_2, \ldots, A_p\}$ be the universal set of attributes. We associate with each attribute $A_i$ of the universe $U$ a countable flat domain consisting of atomic values together with *null*. Each such domain is denoted by $\mathrm{DOM}(A_i)$. For simplicity we assume that the domains $\mathrm{DOM}(A_i) - \{null\}$ are pairwise disjoint.

A *relation scheme* $R$ is a subset of $U$. We extend an atomic domain $\mathrm{DOM}(A)$, where $A \in U$, to a domain over a relation scheme $R \subseteq U$, where $R = \{A_1, A_2, \ldots, A_n\}$, $n \leq p$, as follows:

$$\mathrm{DOM}(R) = \{null\} + (\mathrm{DOM}(A_1) \times \mathrm{DOM}(A_2) \times \ldots \times \mathrm{DOM}(A_n))$$

where $+$ is the disjoint union operator and $\times$ is the Cartesian product operator.

We define a *null-relation*, $r$, over a relation scheme $R \subseteq U$, to be an element of $P(\mathrm{DOM}(R))$, where $P$ is the finite powerset operator. A *tuple* over $R$ is now defined to be an element of a null-relation over $R$. If $r = \{null\}$ then we consider $r$ to be undefined.

EXAMPLE 2.1. Let $R = \{\text{STUDENT, DEPT, MAJOR, CLASS, EXAM, PROJECT}\} = U$ be a relation scheme with the following semantics: a STUDENT is enrolled in a department DEPT and MAJORs in a subject within the department. In addition, a STUDENT attends several CLASSes, each CLASS having several EXAMs during the academic year and one or more PROJECTs. In Figure 2.1 we show a null-relation, $r$, over $R$. We note that we do not attach any specific meaning to *null* appearing in $r$. Thus, *null* for PROJECT, in the fifth tuple, could mean that there does not exist a PROJECT for this *null* STUDENT in the programming CLASS, or it could mean that this *null* STUDENT's PROJECT for this CLASS is unknown.

Let $r$ be a null-relation over $R$. We define *projection* of a tuple $t \in r$ onto $Y \subseteq R$, denoted by $t[Y]$, to be the restriction of $t$ to $Y$. We also refer to $t[Y]$ as the $Y$-value of $t$. We extend the definition of projection to $r$ as follows: $r[Y] = \{t[Y] \mid t \in r\}$.

| STUDENT | DEPT | MAJOR | CLASS | EXAM | PROJECT |
|---|---|---|---|---|---|
| Iris | CS | computing | databases | mid | 1NF |
| Iris | CS | computing | databases | *null* | NF2 |
| Iris | CS | computing | databases | mid | NF2 |
| Iris | CS | computing | databases | *null* | 1NF |
| *null* | CS | computing | programming | final | *null* |
| David | philosophy | logic | first-order | mid | prolog |
| David | *null* | *null* | *null* | final | parlog |
| David | *null* | *null* | first-order | mid | prolog |
| David | philosophy | logic | *null* | final | parlog |
| *null* | philosophy | *null* | *null* | final | *null* |
| *null* | philosophy | *null* | first-order | *null* | functions |

Fig. 2.1. The null-relation $r$.

DEFINITION 2.1. We say that a tuple $t \in r$ is *Y-total*, if $Y \subseteq R$ and $\forall A \in Y$, $t[A]$ is a total value, i.e. $t[A] \in (\mathrm{DOM}(A) - \{\mathrm{null}\})$. We extend the definition of $Y$-total to $r$ as follows: $r$ is $Y$-total if $\forall t \in r$, $t$ is $Y$-total. We call an $R$-total null-relation over $R$ a *total-relation* (or simply a relation).

EXAMPLE 2.2. For the null-relation $r$ shown in Figure 2.1, the first tuple is $R$-total and the fifth tuple is {DEPT, MAJOR, CLASS, EXAM}-total.

We define a *database scheme* over $W \subseteq U$ as a set $\boldsymbol{R} = \{R_1, R_2, \ldots, R_m\}$ such that $\cup_{i=1}^{m} \{R_i\} = W$. In the context of this paper we view a *null-database* as being induced by a null-relation, $r$, over $U$. In particular, a null-database, $d$, over a database scheme $\boldsymbol{R}$, over $W \subseteq U$, is defined by $d = \{r[R_i] \mid R_i \in \boldsymbol{R}\}$.

EXAMPLE 2.3. Let $\boldsymbol{R} = \{\{\mathrm{STUDENT, DEPT, MAJOR}\}, \{\mathrm{STUDENT, CLASS, EXAM}\}, \{\mathrm{STUDENT, CLASS, PROJECT}\}\}$ be a database scheme over $R$ of Example 2.1. Then we can view the null-relation $r$ shown in Figure 2.1 as inducing the null-database $d = \{r[\{\mathrm{STUDENT, DEPT, MAJOR}\}], r[\{\mathrm{STUDENT, CLASS, EXAM}\}], r[\{\mathrm{STUDENT, CLASS, PROJECT}\}]\}$.

Next we extend the definition of the partial order $\leq$ to tuples and then define the *Hoare powerdomain ordering* [18], denoted by $\sqsubseteq$, on null-relations over $R \subseteq U$.

DEFINITION 2.2. Let $R \subseteq U$ be a relation scheme, then

(1) for any tuple $t$ over $R$, *null* $\leq t$ and $t \leq t$.

(2) for any two tuples $t_1$ and $t_2$ over $R$, $t_1 \leq t_2$ if and only if $\forall A \in R$, $t_1[A] \leq t_2[A]$.

If $t_1 \leq t_2$ then we say that $t_1$ is *less informative* than $t_2$ (or equivalently that $t_2$ is *more informative* than $t_1$). A null-relation $r_1$ over $R$ is *less informative* than a null-relation $r_2$ over $R$ (or equivalently $r_2$ is *more informative* than $r_1$), denoted by $r_1 \sqsubseteq r_2$, if and only if $\forall t_1 \in r_1 \, \exists \, t_2 \in r_2$ such that $t_1 \leq t_2$.

We say that a tuple $t_1$ over $R$ is *information-wise equivalent* to a tuple $t_2$ over $R$, denoted by $t_1 \cong t_2$, if and only if $t_1 \leq t_2$ and $t_2 \leq t_1$. Correspondingly, we say that a null-relation $r_1$ over $R$ is *information-wise equivalent* to a null-relation $r_2$ over $R$, also denoted by $r_1 \cong r_2$, if and only if $r_1 \sqsubseteq r_2$ and $r_2 \sqsubseteq r_1$.

EXAMPLE 2.4. Let $t_1, t_2, t_3, t_4$ be the first four tuples in the null-relation, $r$, shown in Figure 2.1. Then, $t_2 \leq t_3$ and $t_4 \leq t_1$. It therefore follows that $(r - \{t_2, t_4\}) \sqsubseteq r$ and $r \sqsubseteq (r - \{t_2, t_4\})$. Thus, $r \cong (r - \{t_2, t_4\})$ holds.

For the rest of the paper we *do not* distinguish between members in each of the equivalence classes of $\cong$ with respect to null-relations. The justification for this approach is that we consider the information content of all null-relations in an equivalence class to be the same.

Next we define the *inequality rule for nulls* and justify our definition.

DEFINITION 2.3. Two values $v_1$ and $v_2$ are equal, i.e. $v_1 = v_2$, if and only if both $v_1$ and $v_2$ are total values.

The above choice of the inequality rule for nulls can be justified as follows: when two null values appearing in a null-relation are updated they may be replaced by two distinct non-null values. We note that due to the inequality rule for nulls our model of a single unmarked null is a special case of the more general model of incomplete information based on multiple marked nulls [9, 10]. In the latter model two null values are considered to be equal if their distinguishing indexes are equal.

We next introduce a more general class of relations, called *or-relations*, which allow a limited form of disjunctive information. Informally, an or-relation over $R \subseteq U$ contains tuples $t$ such that for some $A \in R$, $t[A] \cong v_i \vee \textit{null}$, $v_i$ being a total value; such an $A$-value is termed an *A-or value*. Our motivation for introducing $A$-or values is rather different from that in [7, 10, 15], wherein an $A$-value is extended to a set of possible total values, one of which is the "true" value. $A$-or values in or-relations will allow us to equate two $A$-or values by *promoting* them to equal total values, thus overcoming the limitation of the inequality rule for nulls (see Section 4).

DEFINITION 2.4. We define an *A-or value* to be a disjunction of the form, $v_i \vee \textit{null}$, where $v_i \in (\mathrm{DOM}(A) - \{\textit{null}\})$. We now let $\mathrm{DOM}(A)$ be extended to include $A$-or values, $\forall A \in R$. Thus, an *or-relation* $r$, over a relation scheme $R \subseteq U$, is defined to be an element of $P(\mathrm{DOM}(R))$.

| STUDENT | DEPT | MAJOR | CLASS | EXAM | PROJECT |
|---|---|---|---|---|---|
| David | philosophy | logic | first-order | mid | prolog |
| Raymond $\vee$ *null* | philosophy | math $\vee$ *null* | higher-order $\vee$ *null* | final | *null* |
| Raymond $\vee$ *null* | philosophy | math $\vee$ *null* | first-order | *null* | functions |

Fig. 2.2. The or-relation $r'$.

EXAMPLE 2.5. Let $R$ be the relation scheme given in Example 2.1. An or-relation $r'$ over $R$ is shown in Figure 2.2. We observe that the CLASS-or value, higher-order $\vee$ *null* (for example), induces two sets of null-relations, one in which CLASS is higher-order and the other in which CLASS is *null*. That is, we view the presence of higher-order $\vee$ *null* in $r'$ as a compact representation of the information present in the induced null-relations.

We extend the inequality rule for nulls to include or-values, i.e. two values $v_1$ and $v_2$ (which may be or-values, null values or total values) are equal if and only if both $v_1$ and $v_2$ are total values. We also extend $\sqsubseteq$ to or-relations, where an $A$-total value, $v_i$, is taken to be less informative than an $A$-or value, $v_i \vee$ *null*, i.e. *null* $\leq v_i \leq v_i \vee$ *null* (and $v_i \vee$ *null* $\leq v_i \vee$ *null*). The reason we choose $v_i \leq v_i \vee$ *null* is that we view $\vee$ as union and $\leq$ as set containment. This interpretation of or-values conforms to the Hoare powerdomain ordering. Thus, an $A$-or value $v_i \vee$ *null* appearing in an or-relation, $r$, contains the information that such a value in $r$ represents two sets of null-relations, one which contains $v_i$ and the other which contains *null* (see Example 2.5). It follows that if $r$ contains $n$ distinct or-values, then $r$ is a compact representation of $2^n$ induced null-relations.

We conclude this section with a straightforward proposition, which establishes the connection between the three classes of relations: relations, null-relations and or-relations.

PROPOSITION 2.1. *Let REL(R) be the set of all (total) relations over* $R \subseteq U$, *NULL-REL(R) be the set of all null-relations over R and OR-REL(R) be the set of all or-relations over R. Then*

$$REL(R) \subset NULL\text{-}REL(R) \subset OR\text{-}REL(R).$$

## 3. Null join dependencies and their satisfaction.

In this section, we generalise JDs that hold in (total) relations to *null join dependencies* (NJDs) that hold in null-relations. As a special case of an NJD we have the *null multivalued dependency* (NMVD) [11, 14] when the cardinality of the decomposition is two. We then give the definition of the *inference problem* for NJDs

and formally define when an *inference procedure* for solving the inference problem for NJDs is *sound* and when it is *complete.*

Herein we employ the following useful notation for database schemes found in [2]. Let $\boldsymbol{R}$ and $\boldsymbol{S}$ be two database schemes over $W \subseteq U$. We say that $\boldsymbol{S}$ *covers* $\boldsymbol{R}$, if for every relation scheme, $R_i \in \boldsymbol{R}$, there exists a relation scheme, $S_j \in \boldsymbol{S}$, such that $R_i \subseteq S_j$. The set of all database schemes that cover $\boldsymbol{R}$ is denoted by COVER($\boldsymbol{R}$). In addition, let $\boldsymbol{Q} = \{Q_1, Q_2, \ldots, Q_k\}$ be a database scheme such that $\boldsymbol{Q} \subseteq \boldsymbol{R}$. We say that $\boldsymbol{Q}$ is a *connected subset* of $\boldsymbol{R}$ if and only if there exists a permutation, say $\sigma$, of $\boldsymbol{Q}$ such that $\sigma(Q_i) \cap \sigma(Q_{i+1}) \neq \varnothing$, $1 \leq i < k$.

The database scheme, $\boldsymbol{S}$, is a *covering subset* of the database scheme, $\boldsymbol{R}$, if each relation scheme, $S_i \in \boldsymbol{S}$, is a set of attributes over a connected subset, say $\boldsymbol{S}_i$, of $\boldsymbol{R}$ and such that $\boldsymbol{R} = \cup_i \boldsymbol{S}_i$. We denote the set of all *covering subset database schemes* of $\boldsymbol{R}$ by SUBSET($\boldsymbol{R}$). We note that if $\boldsymbol{S} \in$ SUBSET($\boldsymbol{R}$), then $\boldsymbol{S} \in$ COVER($\boldsymbol{R}$), but the converse is, in general, false.

Let MANY($\boldsymbol{R}$) denote the set of attributes that appear in at least two relation schemes in a database scheme $\boldsymbol{R}$ and let $|\boldsymbol{R}|$ denote the cardinality of $\boldsymbol{R}$.

In the following two definitions we refer to the database schemes, $\boldsymbol{R} = \{R_1, R_2, \ldots, R_m\}$ and $\boldsymbol{S} = \{S_1, S_2, \ldots, S_n\}$ over $W \subseteq U$, with $n \leq m$, and let MANY($\boldsymbol{S}$) $= X$.

DEFINITION 3.1. Let $r$ be a null-relation over $U$. We say that $n$ not necessarily distinct tuples, $t_1, t_2, \ldots, t_n \in r$, are *joinable* on $\boldsymbol{S}$ with the resulting tuple, $t$, over $U$, if the following conditions are true:

(1) $t[S_i \cap X] = t_i[S_i \cap X]$, $1 \leq i \leq n$, i.e. $t[X]$ is $X$-total;
(2) $t[S_i - X] \cong t_i[S_i - X]$, $1 \leq i \leq n$; and
(3) $t[U - W] \cong$ *null.*

We now define the satisfaction of the NJD, $\bowtie[\boldsymbol{R}]$, over $W \subseteq U$, in a null-relation $r$ over $U$.

DEFINITION 3.2. The NJD, $\bowtie[\boldsymbol{R}]$, holds in $r$ if and only if whenever $\exists n$ not necessarily distinct tuples, $t_1, t_2, \ldots, t_n \in r$, that are *joinable* on a covering subset, $\boldsymbol{S} \in$ SUBSET($\boldsymbol{R}$), with the resulting tuple, $t$, over $U$, then $\exists t' \in r$ such that $t \leq t'$ holds.

The NJD, $\bowtie[\boldsymbol{R}]$, is said to be a *trivial* NJD if $m = 1$, otherwise it is said to be a *non-trivial* NJD. An NJD, $\bowtie[\boldsymbol{R}]$, over $W \subseteq U$, is a *full NJD* if $W = U$ and is an *embedded NJD* if $W \subset U$.

It can easily be seen that the NMVD is a special case of the NJD, i.e. when $m = 2$. Also, it can easily be verified that for (total) relations, Definition 3.2 reduces to the standard definition of the JD.

EXAMPLE 3.1. Let $\boldsymbol{R}$ be the database scheme of Example 2.3 and $r$ be the null-relation shown in Figure 2.1. Then it can easily be verified that $r$ satisfies $\bowtie[\boldsymbol{R}]$.

We now define the notion of implication for a set of NJDs. Let $D$ be a set of NJDs, and let SAT(D) denote the set of null-relations, over $U$, that satisfy $D$. We say that $D$ logically implies a single NJD, $d_i$, written in the form $D \models d_i$ if and only if $\mathrm{SAT}(D) \subseteq \mathrm{SAT}(d_i)$.

DEFINITION 3.3. The *inference problem* for NJDs is defined as follows: given a set of NJDs, $D$, and a single NJD, $d_i$, does $D \models d_i$?

An *inference procedure* (for NJDs), $\boldsymbol{P}$, is a (decidable) algorithm for solving the inference problem (for NJDs); $\boldsymbol{P}$ takes as input a set of NJDs, $D$, and a single NJD, $d_i$, and returns true or false. Symbolically $\boldsymbol{P}(D, d_i)$ returns true or false.

DEFINITION 3.4. An inference procedure $\boldsymbol{P}$ is said to be *sound* if whenever $\boldsymbol{P}(D, d_i)$ returns true then $D \models d_i$; $\boldsymbol{P}$ is said to be *complete* if whenever $D \models d_i$ then $\boldsymbol{P}(D, d_i)$ returns true.

We close this section with a brief discussion on whether the sound inference rules for JDs, given in [2], are also sound inference rules for NJDs.

It can be verified that the *covering* and *projection* rules for JDs given in [2] (which are sound for JDs) are also sound for NJDs over null-relations due to Definition 3.2. We note that we can formally show that the covering rule for NJDs holds by using a result from [2], wherein it was shown that a database scheme, $\boldsymbol{S}$, covers a database scheme $\boldsymbol{R}$ if and only if $\boldsymbol{S}$ can be obtained from $\boldsymbol{R}$ by repetitively, either adding a relation scheme to $\boldsymbol{R}$, or by adding an attribute to a relation scheme already in $\boldsymbol{R}$.

On the other hand, the *substitution* rule for JDs also given in [2] (which is sound for JDs) is not, in general, sound for NJDs, since, as is the case with NMVDs, transitivity of NJDs may not hold in the presence of *null* [11, 14]. A simple example illustrates this: let $D = \{AB \twoheadrightarrow C \mid D, A \twoheadrightarrow B \mid CD\}$ be a set of NMVDs (using the standard notation for NMVDs) and $r = \{\langle a_1, null, c_1, d_1\rangle, \langle a_1, null, c_2, d_2\rangle\}$. It can easily be verified that $r \in \mathrm{SAT}(D)$ but $r$ does not satisfy $A \twoheadrightarrow C$, which would be inferred by using the substitution rule. In [13] we present a special case of the substitution rule, called the *non-split substitution rule*, which is sound for NJDs and is sufficient for many practical cases.

## 4. The or-chase procedure for solving the inference problem for NJDs.

In this section we define the *or-chase* procedure as an inference procedure for solving the inference problem for NJDs and as a procedure for testing satisfaction of NJDs in null-relations. The or-chase procedure is an extension of the classical *chase* procedure, which is an inference procedure for solving the inference problem for JDs and a procedure for testing satisfaction of JDs in (total) relations. We denote the result of applying the or-chase procedure to an or-relation, $r$, with respect to a set of

NJDs, $D$, as $ORCHASE_D(r)$. We show that the standard results regarding the classical chase procedure also hold for the or-chase, i.e. $ORCHASE_D(r)$ is finite and unique, and if $r$ is a null-relation then $ORCHASE_D(r)$ results in a null-relation that satisfies $D$. The main result of the paper is then presented, showing that the or-chase procedure is a sound and complete inference procedure for solving the inference problem for NJDs. As a byproduct of our approach the or-chase is also a sound and complete inference procedure for solving the inference problem of embedded NJDs, i.e. the said inference problem is decidable. This is in contrast to the inference problem for embedded JDs which, in general, is known to be only partially decidable. Our stronger result is due to the use of a single unmarked null as opposed to the use of multiple marked nulls in previous formalisms for embedded data dependencies.

We now define the operator TOTAL, which *promotes* $A$-or values to $A$-total values.

DEFINITION 4.1. If $t[A] \cong null$ or $t[A] \cong v_i$ then $\text{TOTAL}(t[A]) \cong t[A]$, otherwise if $t[A] \cong v_i \vee null$ then $\text{TOTAL}(t[A]) \cong v_i$.

TOTAL is extended to $X$-values, where $X = \{A_1, A_2, \ldots, A_n\}$, as follows:
$\text{TOTAL}(t[X]) = \langle \text{TOTAL}(t[A_1]), \text{TOTAL}(t[A_2]), \ldots, \text{TOTAL}(t[A_n]) \rangle$.
If $t[A]$ is the $A$-or value, $v_i \vee null$, then we say that $t[A]$ is *promoted* to $v_i$ by $\text{TOTAL}(t[X])$.

Again in the following two definitions we refer to the database schemes, $\boldsymbol{R} = \{R_1, R_2, \ldots, R_m\}$ and $\boldsymbol{S} = \{S_1, S_2, \ldots, S_n\}$ over $W \subseteq U$, with $n \leq m$, and let $\text{MANY}(\boldsymbol{S}) = X$.

Next we define *or-joinable* tuples in an or-relation, $r$, over $U$, thus generalising the concept of joinable tuples in a null-relation $r$ over $U$.

DEFINITION 4.2. Let $r$ be an or-relation over $U$. We say that $n$ not necessarily distinct tuples, $t_1, t_2, \ldots, t_n \in r$, are *or-joinable* on $\boldsymbol{S}$ with the resulting tuple, $t$, over $U$, if the following conditions are true:

(1) $t[S_i \cap X] = \text{TOTAL}(t_i[S_i \cap X])$, $1 \leq i \leq n$, i.e. $t[X]$ is $X$-total and $\forall A \in (S_i \cap X)$, $t_i[A]$ is either an $A$-total value or an $A$-or value;
(2) $t[S_i - X] \cong t_i[S_i - X]$, $1 \leq i \leq n$; and
(3) $t[U - W] \cong null$.

If $r$ is a null-relation then Definition 4.2 reduces to Definition 3.1.

We now define the *inference rule* (or simply the rule) associated with an NJD, $\bowtie[\boldsymbol{R}] \in D$, where $D$ is a set of NJDs, applied to an or-relation $r$ over $U$; this rule is denoted by $RULE_{\bowtie[\boldsymbol{R}]}(r)$.

DEFINITION 4.3. Let $t_1, t_2, \ldots, t_n \in r$ be $n$ not necessarily distinct tuples that are *or-joinable* on a covering subset, $\boldsymbol{S} \in \text{SUBSET}(\boldsymbol{R})$, with the resulting tuple, $t$, over $U$.

Then the rule associated with the NJD $\bowtie[\boldsymbol{R}]$ applied to $r$ is given by:

$RULE_{\bowtie[\boldsymbol{R}]}(r) = r \cup \{t \mid t$ is the resulting tuple of $n$ or-joinable tuples as defined above$\}$.

Before we define the *or-chase* of an or-relation; $r$, over $U$, with respect to a set of NJDs, $D$, we define the effect on $r$ of invoking *all possible* applications to $r$ of the rules associated with the NJDs in $D$. Thus, we let

$$RULE_D(r) = \{RULE_{d_i}(r) \mid d_i \in D\},$$

i.e. we invoke all possible applications to $r$ of the rules associated with the NJDs $d_i \in D$ in *parallel*, since no ordering is assumed when applying these rules. We note that $RULE_D(r)$ is, by definition, unique.

We next define a transformation function $T_D$, with respect to a set of NJDs, $D$, which operates on an or-relation, $r$, as follows:

$$T_D(r) = \cup \{r' \mid r' \in RULE_D(r)\}.$$

Successive applications of $T_D$ to $r$ yield:

$$T_D^0(r) = r;$$
$$T_D^{i+1}(r) = T_D(T_D^i(r)) \cup T_D^i(r) \text{ with } i = 0, 1, 2, \ldots$$

Finally, we define the *or-chase* of $r$ with respect to a set of NJDs, $D$, denoted by $ORCHASE_D(r)$ (or simply $ORCHASE(r)$ when $D$ is understood from context), as being information-wise equivalent to the *fixpoint* of $r$ with respect to $T_D$, namely

$$ORCHASE_D(r) \cong \cup_{i=0}^{\infty} T_D^i(r).$$

We note that $ORCHASE$ is an *inflationary fixpoint operator* [8], since it can easily be verified that for any or-relation, $r'$, over $U$, $r' \sqsubseteq T_D(r')$. The motivation for using inflationary semantics is that the uniqueness of the or-chase follows directly from its parallel semantics rather than by a standard laborious proof as in [13, 16].

In what follows we refer to a *state* of $ORCHASE(r)$ as an intermediate state thereof, between $r$ and $ORCHASE(r)$, during the computation of $ORCHASE(r)$. The following results establish the fundamental properties of the or-chase procedure (cf. [16]).

LEMMA 4.1. *Let $r$ be an or-relation over $U$ and let $D$ be a set of NJDs. Then $ORCHASE(r)$ is finite and unique.*

PROOF. We first prove that $ORCHASE(r)$ is finite. By definition each application of a rule to a state of $ORCHASE(r)$ incorporates more informative tuples into it, otherwise $ORCHASE(r)$ terminates. Now, let ATOMIC($r$) be the finite set of all values in $r$ together with *null*. We note that rules do not add new values to ATOMIC($r$). Thus, it suffices to show that $ORCHASE(r)$ does not loop forever. Let $r_i$ be the state of $ORCHASE(r)$ after applying some rules to $r$, and let $r_j$ be the state of

$ORCHASE(r)$ after applying some further rules to $r_i$. The result now follows, since $r_i \sqsubseteq r_j$ and $\neg(r_i \cong r_j)$ can only occur a finite number of times and it is also true that $\neg(r_i \cong r_j)$ unless we have $r_i \cong ORCHASE(r)$.

We conclude the proof by showing that $ORCHASE(r)$ is unique. The result now follows since $T_D(r)$ is unique by the definition of $RULE_D(r)$, and $ORCHASE(r)$ results from successively applying the transformation function $T_D$ to the current state of $ORCHASE(r)$. ■

THEOREM 4.2. *Let r be a null-relation over U and let D be a set of NJDs. Then* $ORCHASE_D(r)$ *satisfies D.*

PROOF. $ORCHASE(r)$ satisfies the NJDs in $D$, since the inference rules detect any violation of an NJD in $D$ (see Definition 3.2). It follows that $ORCHASE(r)$ satisfies $D$, otherwise, by the said argument, we can apply one of the rules to $ORCHASE(r)$, thus leading to a contradiction. ■

The following corollary establishes the fact that the or-chase procedure has complete deductive capabilities with respect to NJDs. That is, all possible tuples (and only those tuples) that can be inferred from a null-relation, $r$, with respect to a set of NJDs, $D$, are deduced by $ORCHASE_D(r)$.

COROLLARY 4.3. *Let r be a null-relation over U and let D be a set of NJDs. Then* $r \in SAT(D)$ *if and only if* $ORCHASE_D(r) \cong r$.

PROOF. The result follows from Theorem 4.2 and the definition of the or-chase. ■

In the following we let $a_j \in \mathrm{DOM}(A_j)$ be a *distinguished* total value (cf. *distinguished variable* [19]) associated with the attribute $A_j \in U, j \in \{1, 2, \ldots, p\}$. We now define a *tableau* for an NJD, $\bowtie[\boldsymbol{R}]$, thus generalising the notion of a tableau [2, 16, 19] to or-relations.

DEFINITION 4.4. The *tableau* for the NJD, $\bowtie[\boldsymbol{R}]$, denoted by TABLE($\boldsymbol{R}$), is an or-relation, over $U$, comprising the set of tuples $\{t_1, t_2, \ldots, t_m\}$ such that $t_i$, $1 \leq i \leq m$, is constructed as follows (where $\mathrm{MANY}(\boldsymbol{R}) = X$):

(1) $\forall A_j \in (R_i \cap X), t_i[A_j] \cong a_j \vee \textit{null}$;
(2) $\forall A_j \in (R_i - X), t_i[A_j] \cong a_j$; and
(3) $\forall A_j \in (U - R_i), t_i[A_j] \cong \textit{null}$.

EXAMPLE 4.1. Let $\boldsymbol{R} = \{\mathrm{ABC}, \mathrm{ABD}, \mathrm{AE}\}$, then the tableau for $\boldsymbol{R}$, TABLE($\boldsymbol{R}$), is shown in Figure 4.1. Also, let $\boldsymbol{R}' = \{\mathrm{AB}, \mathrm{BCD}, \mathrm{CE}\}$, then the tableau for $\boldsymbol{R}'$, TABLE($\boldsymbol{R}'$), is shown in Figure 4.2.

| A | B | C | D | E |
|---|---|---|---|---|
| $a_1 \vee null$ | $a_2 \vee null$ | $a_3$ | *null* | *null* |
| $a_1 \vee null$ | $a_2 \vee null$ | *null* | $a_4$ | *null* |
| $a_1 \vee null$ | *null* | *null* | *null* | $a_5$ |

Fig. 4.1. TABLE ({ABC, ABD, AE}).

| A | B | C | D | E |
|---|---|---|---|---|
| $a_1$ | $a_2 \vee null$ | *null* | *null* | *null* |
| *null* | $a_2 \vee null$ | $a_3 \vee null$ | $a_4$ | *null* |
| *null* | *null* | $a_3 \vee null$ | *null* | $a_5$ |

Fig. 4.2. TABLE ({AB, BCD, CE}).

Next we define the valuation [3, 16, 19] of a tableau.

DEFINITION 4.5. Let $\phi$ be a mapping (called a *valuation*), which maps a tableau into a set of null-relations over $U$ as follows: $\phi$ maps a total value $a_i$ to $a_i$, maps *null* to *null* and maps an or-value, $a_i \vee null$, either to $a_i$ or to *null*. Correspondingly for a tuple $\langle v_1, v_2, \ldots, v_p \rangle$ we have: $\phi(\langle v_1, v_2, \ldots, v_p \rangle) = \langle \phi(v_1), \phi(v_2), \ldots, \phi(v_p) \rangle$; $\phi$ is extended to a tableau $T$ as follows: $\phi(T) = \{\phi(t) \mid t \in T\}$.

The set of null-relations induced by the mappings $\phi$, denoted as NULL($T$), is defined by: NULL($T$) = $\{r \mid \exists \phi$ such that $\phi(T) \cong r\}$. The set of valuations induced by NULL($T$), denoted as $\Phi(T)$, is defined by: $\Phi(T) = \{\phi \mid \phi(T) \in \text{NULL}(T)\}$. We observe that if $T$ contains $n$ distinct or-values then the cardinality of NULL($T$) and therefore $\Phi(T)$ is $2^n$.

In the results that follow we utilise the five abbreviations of the expressions numbered (1) to (5) given below; $\boldsymbol{R}$ is a database scheme over $W \subseteq U$, $D$ is a set of NJDs and $\phi \in \Phi(\text{TABLE}(\boldsymbol{R}))$.

(1) $ORCHASE_D(\phi(\text{TABLE}(\boldsymbol{R})))$ is abbreviated to $OR_D(\phi(T))$.
(2) $ORCHASE_{D \cup \{\bowtie[\boldsymbol{R}]\}}(\phi(\text{TABLE}(\boldsymbol{R})))$ is abbreviated to $OR_{D \cup \boldsymbol{R}}(\phi(T))$.
(3) $\phi(ORCHASE_D(\text{TABLE}(\boldsymbol{R})))$ is abbreviated to $\phi(OR_D(T))$.
(4) $ORCHASE_D(\text{TABLE}(\boldsymbol{R}))$ is abbreviated to $OR_D(T)$.
(5) $ORCHASE_{D \cup \{\bowtie[\boldsymbol{R}]\}}(\text{TABLE}(\boldsymbol{R}))$ is abbreviated to $OR_{D \cup \boldsymbol{R}}(T)$.

The following lemma shows what happens when a valuation, $\phi$, is interchanged with the or-chase procedure.

LEMMA 4.4. *Let $\boldsymbol{R}$ be a database scheme over $W \subseteq U$ and let $D$ be a set of NJDs. Then $\forall \phi \in \Phi(TABLE(\boldsymbol{R}))$, $OR_D(\phi(T)) \sqsubseteq \phi(OR_D(T))$.*

PROOF. Let $\phi \in \Phi(\text{TABLE}(\boldsymbol{R}))$ be a valuation. We prove the result by induction on the application of the rules during the successive applications of $T_D$ to $\phi(\text{TABLE}(\boldsymbol{R}))$ and $\text{TABLE}(\boldsymbol{R})$.

BASIS. Consider any tuple $t \in \phi(\text{TABLE}(\boldsymbol{R}))$, then $t \in OR_D(\phi(T))$ and also $\exists\, t' \in \phi(OR_D(T))$ such that $t \leq t'$, trivially.

INDUCTION. *Let $\hat{r}_1$* be an intermediate state of the tableau during the computation of $OR_D(\phi(T))$ and let $\hat{r}_2$ be an intermediate state of the tableau during the computation of $OR_D(T)$ such that $\hat{r}_1 \sqsubseteq \phi(\hat{r}_2)$. Now, if $t_1, t_2, \ldots, t_n \in \hat{r}_1$ are joinable tuples over a covering subset database scheme $\boldsymbol{S}$ of $\boldsymbol{R}$ with the resulting tuple $t \in OR_D(\phi(T))$, then by inductive hypothesis there must exist tuples $t'_1, t'_2, \ldots, t'_n \in \hat{r}_2$, such that $t_1 \leq \phi(t'_1)$, $t_2 \leq \phi(t'_2), \ldots, t_n \leq \phi(t'_n)$. Since $t_1, t_2, \ldots, t_n$ are joinable on $\boldsymbol{S}$ then so are $\phi(t'_1), \phi(t'_2), \ldots, \phi(t'_n)$ with the resulting tuple $\phi(t')$; consequently $t'_1, t'_2, \ldots, t'_n$ are or-joinable on $\boldsymbol{S}$ with the resulting tuple $t' \in OR_D(T)$ and thus $\phi(t') \in \phi(OR_D(T))$. The result now follows that $t \leq \phi(t')$ as required, since or-joinability is a monotone mapping. ■

We note that the reverse of Lemma 4.4, i.e. $\phi(OR_D(T)) \sqsubseteq OR_D(\phi(T))$ is, in general, false. For example, let $D = \{\bowtie[\boldsymbol{R}]\}$ where $\boldsymbol{R} = \{\text{ABC}, \text{ABD}, \text{AE}\}$ as in Example 4.1, and let $\phi$ map $a_1 \vee \textit{null}$ to $a_1$ and $a_2 \vee \textit{null}$ to *null*. Then, it can easily be verified that the tuple, $\langle a_1, a_2, a_3, a_4, a_5 \rangle$, is in $\phi(OR_D(T))$ but not in $OR_D(\phi(T))$.

The following lemma establishes the relationship between the or-chase of a tableau, say $T$ (which is an or-relation), and the or-chase of its set of corresponding null-relations, $\text{NULL}(T)$.

LEMMA 4.5. *Let $\boldsymbol{R}$ be a database scheme over $W \subseteq U$ and let $D$ be a set of NJDs. Then $OR_D(T) \cong OR_{D \cup \boldsymbol{R}}(T)$ if and only if $\forall \phi \in \Phi(TABLE(\boldsymbol{R}))$, $OR_D(\phi(T)) \cong OR_{D \cup \boldsymbol{R}}(\phi(T))$.*

PROOF. (*IF*): Let $\phi \in \Phi(\text{TABLE}(\boldsymbol{R}))$ be a valuation. We prove the result by contradiction to the fact that $\forall \phi \in \Phi(\text{TABLE}(\boldsymbol{R}))$; $OR_{D \cup \boldsymbol{R}}(\phi(T)) \sqsubseteq OR_D(\phi(T))$. Now, since $OR_D(T) \sqsubseteq OR_{D \cup \boldsymbol{R}}(T)$ holds trivially we assume that $\neg(OR_{D \cup \boldsymbol{R}}(T) \sqsubseteq OR_D(T))$. Thus, there must exist a tuple, $t$, such that $t \in OR_{D \cup \boldsymbol{R}}(T)$ and there *does not exist* a tuple $t'$, where $t \leq t'$ and $t[\text{MANY}(\boldsymbol{R})] \cong t'[\text{MANY}(\boldsymbol{R})]$, such that $t' \in OR_D(T)$.

We can now choose a valuation $\phi \in \Phi(\text{TABLE}(\boldsymbol{R}))$ such that $\phi$ promotes an $A$-or value, $a_i \vee \textit{null}$, to an $A$-total value $a_i$ if and only if $t$ is $A$-total and $A \in \text{MANY}(\boldsymbol{R})$. Let $\hat{r}_1$ be an intermediate state during the computation of $OR_{D \cup \boldsymbol{R}}(T)$ and let $\hat{r}_2$ be an intermediate state during the computation of $OR_{D \cup \boldsymbol{R}}(\phi(T))$. By the choice of $\phi$ we

observe that $t \in OR_{D\cup R}(T)$ implies $\phi(t) \in OR_{D\cup R}(\phi(T))$. This is due to the fact that if $\exists \hat{r}_1$ such that $t_1, t_2, \ldots, t_n \in \hat{r}_1$ are or-joinable tuples over a covering subset database scheme $S$ of $R$ with the resulting tuple $t \in OR_{D\cup R}(T)$, then $\exists \hat{r}_2$ such that $\phi(t_1), \phi(t_2), \ldots, \phi(t_n) \in \hat{r}_2$ are joinable tuples over $S$ with the resulting tuple $\phi(t) \in OR_{D\cup R}(\phi(T))$ (this can be proved by induction). Similarly, $t' \notin OR_D(T)$ implies $\phi(t') \notin OR_D(\phi(T))$, since if $\nexists \hat{r}_3$, an intermediate state during the computation of $OR_D(T)$, such that a set of or-joinable tuples in $\hat{r}_3$ would result in the tuple $t' \in OR_D(T)$, then $\nexists \hat{r}_4$, an intermediate state during the computation of $OR_D(\phi(T))$, such that a corresponding set of joinable tuples in $\hat{r}_4$ would result in the tuple $\phi(t') \in OR_D(\phi(T))$. From the above we obtain a contradiction to the fact that $OR_{D\cup R}(\phi(T)) \sqsubseteq OR_D(\phi(T))$, thus proving the result.

*(ONLY IF)*: We prove the result by contradiction to the fact that $OR_D(T) \sqsubseteq OR_{D\cup R}(T)$. Now, since $OR_D(\phi(T)) \sqsubseteq OR_{D\cup R}(\phi(T))$ holds trivially we assume that $\neg(OR_{D\cup R}(\phi(T)) \sqsubseteq OR_D(\phi(T)))$. Thus, there must exist a valuation $\phi \in \Phi(\text{TABLE}(R))$ and a tuple, $u$, such that $u \in OR_{D\cup R}(\phi(T))$ and there *does not exist* a tuple, $u'$, where $u \leq u'$ and $u[\text{MANY}(R)] \cong u'[\text{MANY}(R)]$, such that $u' \in OR_D(\phi(T))$.

We can now choose a tuple $t \in OR_{D\cup R}(T)$ such that $t[A]$ is an $A$-or value or an $A$-total value if and only if $u = \phi(t)$ is $A$-total and $A \in \text{MANY}(R)$. By the choice of $t$ we conclude using an argument similar to that in the if part that $\phi(t) \in OR_{D\cup R}(\phi(T))$ implies $t \in OR_{D\cup R}(T)$. Similarly, $u' = \phi(t') \notin OR_D(\phi(T))$ implies $t' \notin OR_D(T)$. From the above we obtain a contradiction to the fact that $OR_{D\cup R}(T) \sqsubseteq OR_D(T)$, thus proving the result. ■

The ensuing theorem establishes the main result of the paper, namely that the or-chase procedure solves the inference problem for NJDs.

THEOREM 4.6. *Let $D$ be a set of NJDs and let $\bowtie[R]$ be a single NJD. Then $D \models \bowtie[R]$ if and only if $OR_D(T) \cong OR_{D\cup R}(T)$.*

PROOF. *(IF)*: If $OR_D(T) \cong OR_{D\cup R}(T)$ then by the only if part of Lemma 4.5, $\forall \phi \in \Phi(\text{TABLE}(R))$, $OR_D(\phi(T)) \cong OR_{D\cup R}(\phi(T))$. Now, let $r \in \text{SAT}(D)$ be a null-relation over $U$ and let $t_1, t_2, \ldots, t_n \in r$ be joinable tuples over a covering subset database scheme $S$ of $R$, with the resulting tuple, $t$, over $U$. We now define a mapping $\hat{\phi}$ for the tuples $t_i$, $1 \leq i \leq n$, as follows: $\hat{\phi}$ maps *null* to *null* and maps a total value $t[A_j]$ to the distinguished total value $a_j \in \text{DOM}(A_j)$ for each $A_j \in U$. From the definition of $\hat{\phi}$ it follows that $\exists \phi \in \Phi(\text{TABLE}(R))$ such that $t_i' \leq \hat{\phi}(t_i)$, where $t_i' \in OR_D(\phi(T))$ and $t_i'[\text{MANY}(R)] = \hat{\phi}(t_i[\text{MANY}(R)])$, $1 \leq i \leq n$. Now, since $t_i$ are joinable over $S$, then $t_i'$, $1 \leq i \leq n$, are also joinable over $S$ with the resulting tuple $t' \in OR_D(\phi(T))$. It follows that $t \in r$ since $t' \leq \hat{\phi}(t)$ and $OR_D(\phi(T)) \cong OR_{D\cup R}(\phi(T))$, implying that $r \in \text{SAT}(D \cup \{\bowtie[R]\})$ as required.

*(ONLY IF)*: If $D \models \bowtie[R]$ then $\text{SAT}(D \cup \{\bowtie[R]\}) = \text{SAT}(D)$. Thus, $\forall r \in \text{NULL}(\text{TABLE}(R))$, $ORCHASE_D(r) \in \text{SAT}(D \cup \{\bowtie[R]\})$, since by Theorem 4.2 $ORCHASE_D(r) \in \text{SAT}(D)$. It follows from Lemma 4.1 and Corollary 4.3 that

$\forall r \in \text{NULL}(\text{TABLE}(\boldsymbol{R}))$, $ORCHASE_D(r) \cong ORCHASE_{D \cup \{\bowtie[\boldsymbol{R}]\}}(r)$. Thus, the result follows by the if part of Lemma 4.5. ■

We note that the above results are also obtained for embedded NJDs, since in our formalism embedded NJDs are a special case of NJDs, i.e. when $W \subset U$. It is interesting to observe that, when using marked nulls, testing implication for embedded data dependencies (embedded JDs being a special case) is, in general, known to be only partially decidable. As a consequence of Theorem 4.6, in our context, the or-chase procedure is decidable for embedded NJDs as well as for NJDs.

The following corollary follows directly from Theorem 4.6.

COROLLARY 4.7. *Let* $\boldsymbol{P}$ *be an inference procedure for NJDs, which returns true if and only if* $OR_D(T) \cong OR_{D \cup \boldsymbol{R}}(T)$. *Then* $\boldsymbol{P}$ *is a sound and complete inference procedure for NJDs.*

EXAMPLE 4.2. Let $\boldsymbol{R} = \{\text{ABC}, \text{ABD}, \text{AE}\}$ and $D = \{\bowtie[\{\text{ABC}, \text{ABDE}\}], \bowtie[\{\text{ABD}, \text{AE}\}]\}$. Then it can be verified that $\neg(ORCHASE_D(\text{TABLE}(\boldsymbol{R})) \cong ORCHASE_{D \cup \{\bowtie[\boldsymbol{R}]\}}(\text{TABLE}(\boldsymbol{R})))$, where $ORCHASE_{D \cup \{\bowtie[\boldsymbol{R}]\}}(\text{TABLE}(\boldsymbol{R}))$ is shown in Figure 4.3, since the tuple $t \cong \langle a_1, a_2 \vee \textit{null}, a_3, \textit{null}, a_5 \rangle \in ORCHASE_{D \cup \{\bowtie[\boldsymbol{R}]\}}(\text{TABLE}(\boldsymbol{R}))$ but $t \notin ORCHASE_D(\text{TABLE}(\boldsymbol{R}))$, implying $D \not\models \bowtie[\boldsymbol{R}]$.

Example 4.2 illustrates the fact that unlike the classical chase procedure, in the or-chase procedure a tuple of distinguished total values does *not* necessarily indicate NJD implication.

EXAMPLE 4.3. Let $\boldsymbol{R}' = \{\text{AB}, \text{BCD}, \text{CE}\}$ and $D = \{\bowtie[\{\text{AB}, \text{BCDE}\}], \bowtie[\{\text{CE}, \text{ABCD}\}]\}$. Then it can be verified that $ORCHASE_D(\text{TABLE}(\boldsymbol{R}')) \cong ORCHASE_{D \cup \{\bowtie[\boldsymbol{R}']\}}(\text{TABLE}(\boldsymbol{R}'))$, as shown in Figure 4.4, implying $D \models \bowtie[\boldsymbol{R}']$.

Some of the tuples, which are less informative than the tuples appearing in Figures 4.3 and 4.4, respectively, are omitted. This does not affect the result, since we do not distinguish between members in each of the equivalence classes induced by $\cong$.

## 5. Concluding remarks.

We have defined a class of data dependencies, namely NJDs, which hold in null-relations, and have shown that the or-chase is a sound and complete inference procedure for NJDs. In order to prove this result we generalised null-relations to or-relations by allowing them to contain a limited form of disjunctive information. This result implies that the inference problem for NJDs is decidable; this is due to the fact that in our model for incomplete information we include only a single unmarked null value, i.e. *null*. A further implication of our result is: if $r$ contains $n$ distinct or-values then $ORCHASE(r)$ is equivalent to applying the or-chase procedure to the

| A | B | C | D | E |
|---|---|---|---|---|
| $a_1 \vee null$ | $a_2 \vee null$ | $a_3$ | $null$ | $null$ |
| $a_1 \vee null$ | $a_2 \vee null$ | $null$ | $a_4$ | $null$ |
| $a_1 \vee null$ | $null$ | $null$ | $null$ | $a_5$ |
| $a_1$ | $a_2$ | $a_3$ | $a_4$ | $null$ |
| $a_1$ | $a_2$ | $null$ | $a_4$ | $a_5$ |
| $a_1$ | $a_2$ | $a_3$ | $a_4$ | $a_5$ |
| $a_1$ | $a_2 \vee null$ | $null$ | $a_4$ | $a_5$ |
| $a_1$ | $a_2 \vee null$ | $a_3$ | $null$ | $a_5$ |

Fig. 4.3. $ORCHASE_{D \cup \{\bowtie[R]\}}(\text{TABLE}(R))$.

| A | B | C | D | E |
|---|---|---|---|---|
| $a_1$ | $a_2 \vee null$ | $null$ | $null$ | $null$ |
| $null$ | $a_2 \vee null$ | $a_3 \vee null$ | $a_4$ | $null$ |
| $null$ | $null$ | $a_3 \vee null$ | $null$ | $a_5$ |
| $a_1$ | $a_2$ | $a_3 \vee null$ | $a_4$ | $null$ |
| $a_1$ | $a_2$ | $a_3$ | $a_4$ | $a_5$ |
| $null$ | $a_2 \vee null$ | $a_3$ | $a_4$ | $a_5$ |

Fig. 4.4. $ORCHASE_D(\text{TABLE}(R')) \cong ORCHASE_{D \cup \{\bowtie[R']\}}(\text{TABLE}(R'))$.

$2^n$ null-relations induced by $r$.

Finally, our results are more general than the ones presented in [11], since we make no assumptions about the acyclicity or cyclicity of the database scheme as is the case therein, where only $\gamma$-acyclic database schemes [5] are considered.

**Acknowledgement.**

The referees' comments are greatly appreciated.

## REFERENCES

1. P. Atzeni and M. C. Bernardis, *A new interpretation for null values in the weak instance model*, Jour. Comput. Syst. Sci., Vol. 41, pp. 25–43, 1990.
2. C. Beeri and M. Y. Vardi, *On the properties of join dependencies, Advances in Database Theory*, Vol. 1, H. Gallaire, J. Minker and J. M. Nicholas, Eds., Plenum Press, New York, 1981, pp. 25–72.

3. C. Beeri and M. Y. Vardi, *A proof procedure for data dependencies*, Jour. ACM, Vol. 31, pp. 718–741, 1984.
4. E. F. Codd, *The Relation Model for Database Management: Version 2*, Addison Wesley, Reading, MA., 1990.
5. R. Fagin, *Degrees of acyclicity for hypergraphs and relational database systems*, Jour. ACM, Vol. 30, pp. 514–550, 1983.
6. M. H. Graham, A. O. Mendelzon and M. Y. Vardi, *Notions of dependency satisfaction*, Jour. ACM, Vol. 33, pp. 105–129, 1986.
7. J. Grant, *Incomplete information in a relational database*, Fundamenta Informaticae, Vol. 3, pp. 363–378, 1980.
8. Y. Gurevich and S. Selah, *Fixed-point extensions of first-order logic*, Annals of Pure and Applied Logic, Vol. 32, pp. 265–280, 1986.
9. T. Imielinski and W. Lipski Jr., *Incomplete information in relational databases*, Jour. ACM, Vol. 31, pp. 761–791, 1984.
10. T. Imielinski, *Incomplete information in logical databases*, IEEE Quarterly Bulletin on Data Engineering, Vol. 12, pp. 29–40, 1989.
11. S. Jajodia and F. N. Springsteel, *Lossless outer joins with incomplete information*, BIT, Vol. 30, pp. 34–41, 1990.
12. N. Lerat and W. Lipski Jr., *Nonapplicable nulls*, Theoret. Comput. Sci., Vol. 46, pp. 67–82, 1986.
13. M. Levene, *The Nested Universal Relation Database Model*, Lecture Notes in Computer Science, Vol. 595, Springer-Verlag, Berlin, 1992.
14. Y. E. Lien, *Multivalued dependencies with null values in relational databases*, Proc. of 5th Conf. on Very Large Data Bases, Rio de Janeiro, Brazil, pp. 61–66, 1979.
15. K.-C. Liu and R. Sunderraman, *Indefinite and maybe information in relational databases*, ACM Trans. on Database Syst., Vol. 15, pp. 1–39, 1990.
16. D. Maier, A. O. Mendelzon and Y. Sagiv, *Testing implication of data dependencies*, ACM Trans. on Database Syst., Vol. 4, pp. 455–469, 1979.
17. Y. Sagiv and S. F. Walecka, *Subset dependencies and a completeness result for a subclass of embedded multivalued dependencies*, Jour. ACM, Vol. 29, pp. 103–117, 1982.
18. D. A. Schmidt, *Denotational Semantics: A Methodology for Language Development*, Allyn and Beacon, Inc., Newton, MA., 1986.
19. E. Sciore, *A complete axiomatization of join dependencies*, Jour. ACM, Vol. 29, pp. 373–393, 1982.
20. J. Stein and D. Maier, *Relaxing the universal relation scheme assumption*, Proc. of the 4th ACM SIGACT-SIGMOD Symposium on Principles of Database Systems, Portland, OR., ACM, New York, 1985, pp. 76–85.
21. J. D. Ullman, *Principles of Database and Knowledge-Base Systems*, Vol. 1, Computer Science Press, Rockville, Maryland, 1988.
22. C. Zaniolo, *Database relations with null values*, Jour. Comput. Syst. Sci., Vol. 28, pp. 142–166, 1984.