CHAPTER - 2

FUNCTIONAL AND MULTIVALUED DEPENDENCIES WITH NULLS

2.1 Introduction

In this chapter, we consider briefly the three classes of (classical) data dependencies, viz., the class of functional dependencies, the class of multivalued dependencies, and the class of functional and multivalued dependencies. A sound and complete formal system is noted for each of the above class of constraints. The solution to the membership problem: Given a set of dependencies $\Gamma$ and a dependency $f$, is $f$ implied by $\Gamma$? is noted for each of the above classes. We have sketched several notions of covers of functional/multivalued dependencies. Then we extend the classical relational database model to incorporate nulls with "no-information" interpretation. In this extended model, the above three classes of data dependencies are generalized to respectively, the class of functional dependencies with nulls, the class of multivalued dependencies with nulls, and the class of functional and multivalued dependencies. For each of the classes, we list a sound and complete system and give algorithms to solve the membership problem.

2.2 On Functional and Multivalued Dependencies.

As remarked in the preceeding chapter, functional dependencies and multivalued dependencies are central to the problem of designing better normalized database scheme; the validity of the outcome of the various decomposition/synthesis algorithms heavily depend on the answer to the question "Do we
know all relevant dependencies?". It is this question which lies at the heart of this thesis.

The existence of a complete formal system for a set \( \Gamma \) of constraints ensures that we can derive, by means of the inference rules for the constraints, all the constraints implied by \( \Gamma \). Given a set of dependencies, it is possible to derive additional dependencies from it. The inference rules for functional dependencies were studied by Armstrong [5]. The inference rules for multivalued dependencies and/or multivalued dependencies together with functional dependencies were studied in [14, 33, 109]. The rules we present here are from [14].

We distinguish three groups of rules. The first group contains rules for functional dependencies, that is, rules that allow us to derive functional dependencies from given functional dependencies. The second group contains rules that allow us to derive multivalued dependencies from given multivalued dependencies. The third group is the group of rules for both types of dependencies. It contains the first two groups, and also the set of mixed rules, that allow us to derive dependencies when functional and multivalued dependencies together are given.

For all groups, it is not our purpose to list all the complete set of inference rules. We list several inference rules and a complete set of inference rules for each group. For proofs of soundness of the rules see [5, 14]. In the rules \( X, Y, Z, W \), are subsets of a relation scheme \( R \).
Functional Dependency Rules

F1 (Reflexivity). If \( Y \subseteq X \), then \( X \rightarrow Y \) holds.

F2 (Augmentation). If \( X \rightarrow Y \) holds and \( Z \subseteq W \), then \( XW \rightarrow YZ \) holds.

F3 (Transitivity). If \( X \rightarrow Y \) and \( Y \rightarrow Z \) holds, then \( X \rightarrow Z \) holds.

F4 (Pseudotransitivity). If \( X \rightarrow Y \) and \( YW \rightarrow Z \) holds, then \( XW \rightarrow Z \) holds.

F5 (Union). If \( X \rightarrow Y \) and \( X \rightarrow Z \) hold, then \( X \rightarrow YZ \) holds.

F6 (Decomposition). If \( X \rightarrow YZ \) holds, then \( X \rightarrow Y \) and \( X \rightarrow Z \) holds.

In this group F1-F3 are sufficient, since the rules F4-F6 are implied by them.

Multivalued Dependency Rules

M0 (Complementation). Let \( X,Y \) and \( Z \) be sets such that \( XYZ = R \) and \( Y \cap Z \subseteq X \), then \( X \rightarrow Y \rightarrow Z \) holds if and only if \( X \rightarrow Z \) holds.

M1 (Reflexivity). If \( Y \subseteq X \), then \( X \rightarrow Y \rightarrow Z \) holds.

M2 (Augmentation). If \( X \rightarrow Y \rightarrow Z \) holds and \( Z \subseteq W \), then \( XW \rightarrow YZ \) holds.

M3 (Transitivity). If \( X \rightarrow Y \rightarrow Z \) hold, then \( X \rightarrow Z \rightarrow Y \) holds.

M4 (Pseudotransitivity). If \( X \rightarrow Y \rightarrow Z \) hold, then \( XW \rightarrow YZ \rightarrow W \) holds.

M5 (Union). If \( X \rightarrow Y \rightarrow Z \) and \( X \rightarrow Y \rightarrow Z \) hold, then \( X \rightarrow YZ \) holds.

M6 (Decomposition). If \( X \rightarrow Y \rightarrow Z \) hold, then \( X \rightarrow Y \rightarrow Z \), \( X \rightarrow Y \rightarrow Z \rightarrow Z \), \( X \rightarrow Y \rightarrow Z \rightarrow Z \), and \( X \rightarrow Z \rightarrow Y \) hold.

For this group, the rules M0-M3 are sufficient; the rules M4-M6 are implied by them. In the third group, there are two additional rules.

FM1 If \( X \rightarrow Y \) holds, then \( X \rightarrow Y \rightarrow Z \) holds.

FM2 If \( X \rightarrow Y \rightarrow Z \) and \( Y \rightarrow Z \) hold, then \( X \rightarrow Z \rightarrow Y \) holds.
The partition of the rules into three groups reflects the fact that we are dealing with three families (sets) of dependencies: the family of functional dependencies, the family of multivalued dependencies, and the family of functional and multivalued dependencies.

From each of the group of inference rules, we can select a complete set of inference rules. In order to do that, we, first, introduce some preliminary definitions. In the following $F$ denotes a set of functional dependencies, $M$ denotes a set of multivalued dependencies, and $FuM$ denotes the set of functional dependencies from $F$ and multivalued dependencies from $M$.

A functional dependency $X \rightarrow Y$ can be derived from $F$ (resp. from $FuM$), written $F \vdash X \rightarrow Y$ (resp. $FuM \vdash X \rightarrow Y$), if $X \rightarrow Y \in F$ (resp. $X \rightarrow Y \in FuM$), or $X \rightarrow Y$ can be obtained from $F$ (resp. from $FuM$) by a finite number of applications of the inference rules $F1-F3$ (resp. $F1-F3$, $M0-M3$, $FM1$, and $FM2$).

A multivalued dependency $X \leftrightarrow Y$ can be derived from $M$ (resp. from $FuM$), written $M \vdash X \leftrightarrow Y$ (resp. $FuM \vdash X \leftrightarrow Y$), if $X \leftrightarrow Y \in M$ (resp. $X \leftrightarrow Y \in FuM$), or $X \leftrightarrow Y$ can be obtained from $M$ (resp. from $FuM$) by a finite number of application of the inference rules $M0-M3$ (resp. $F1-F3$, $M0-M3$, $FM1$, and $FM2$).

A functional dependency $X \rightarrow Y$ is implied by $F$ (resp. by $FuM$) if in every relation in which all the dependencies of $F$ (resp. of $FuM$) hold, $X \rightarrow Y$ also holds. By $F^+$ we denote the set of all dependencies implied by $F$.

A multivalued dependency $X \leftrightarrow Y$ is implied by $M$ (resp. $FuM$) if in every relation in which all the dependencies of $M$ (resp.
of $\text{FUM}$ hold, $X \rightarrow Y$ also holds. By $M^+$ (resp. $(\text{FUM})^+$, we denote the set of all dependencies implied by $M$ (resp. $\text{FUM}$).

The following theorem states the soundness and completeness of the various formal systems.

2.2.1 Theorem: Let $F$ be a set of functional dependencies and $M$ be a set of multivalued dependencies. Then

(a) $[5,14] + : X \rightarrow Y \in F^+ \iff F \vdash X \rightarrow Y.$
(b) $[14] + : X \leftrightarrow Y \in M^+ \iff M \vdash X \leftrightarrow Y.$
(c) $[14] + : X \rightarrow Y \in (\text{FUM})^+ \iff \text{FUM} \vdash X \rightarrow Y.$
(d) $[14] + : X \leftrightarrow Y \in (\text{FUM})^+ \iff \text{FUM} \vdash X \leftrightarrow Y.$

If it turns out that computing $\Gamma^+$ of a set $\Gamma$ of dependencies is a time consuming task in general, simply because the set of dependencies $\Gamma^+$ can be large even if $\Gamma$ is small. For instance, if $\Gamma = \{A \rightarrow B_1, A \rightarrow B_2, ..., A \rightarrow B_n\}$ be a set of functional dependencies. Then $\Gamma^+$ includes all the dependencies $A \rightarrow Y$, where $Y$ is a subset of $\{B_1, B_2, ..., B_n\}$ as there are $2^n$ such sets $Y$, we could not expect to list $\Gamma^+$ conveniently, even for reasonably sized $n[101]$. Fortunately, for most purposes, we need only tell whether a dependency $f$ is in $\Gamma^+$ or not. Thus, if $\Gamma$ is equal to $F$, a set of functional dependencies, and $f$ be a functional dependency, then the membership problem is to tell whether $f \in F^+$. The membership problem for functional dependencies is solved in [12]; for this we, first, define the closure of a set of attributes $X$ with respect to a set of functional dependencies $F$, written as $X_F^+$, as follows:

$$X_F^+ = \{A : X \rightarrow A \in F^+\}$$

A very important lemma for functional dependencies [101] states as follows:
2.2.2 Lemma: Let $F$ be a set of functional dependencies. Then $F \models X \rightarrow Y$ if and only if $Y \subseteq X_F^+$.

Computing $X_F^+$, for a set of attributes is not hard as will be seen next. By Lemma 2.2.2, telling whether $X \rightarrow Y$ in $F^+$ is no harder than computing $X_F^+$. The following algorithm from [12] computes the closure of a set of attributes $X$ with respect to a set of functional dependencies $F$.

2.2.3 Algorithm

```
procedure CLOSURE (X,F)
begin
    CLOSURE ← X
    OLDCLOSURE ← ∅
    while CLOSURE ≠ OLDCLOSURE do
    begin
        OLDCLOSURE ← CLOSURE
        for each $V \rightarrow W$ in $F$ do
            if $V \subseteq CLOSURE$ then CLOSURE ← CLOSURE ∪ W
        end
    end
    In the worst case, the above algorithm executes the external loop once for each attribute and the internal once for each functional dependency. Therefore, if $X \subseteq R$, $|R| = m$, $|F| = m_1$, and we follow the $m$ bit representation of subsets of $R[2]$, then in the internal loop, checking whether $V \subseteq CLOSURE$ takes the time $O(m)$; similarly, taking the union of two sets also requires time $O(m)$. Therefore, the internal loop can be executed in time $O(m_1 \cdot m)$. Thus, the total time spent, in the worst case, by the above algorithm is $O(m_1 \cdot m^2)$. The
```
correctness of the algorithm is proved showing that $X_F^+$ contains and is contained in the set of attributes that is value of the variable CLOSURE at the end of the execution of the algorithm. Both parts are proved by induction, one on the number of steps executed by the algorithm, and the other one on the length of a derivation of $X\rightarrow X_F^+$ from $X$. At this point, the membership algorithm is immediate, and it also runs in time $O(m_1 m^2)$: given $F$ and $X \rightarrow Y$, it first, computes $X_F^+$, and then checks whether $Y \subseteq X_F^+$.

The membership problem for multivalued dependencies is the following: Given a set of $M$ of multivalued dependencies, and a multivalued dependency $g$, is $g$ in $M^+$? Now, we present a polynomial time algorithm from Beeri [12] that solves this problem. First, we introduce the notion of dependency basis of a set of attributes with respect to a set of multivalued dependencies as follows.

A set $\{X_1, X_2, \ldots, X_p\}$ of non-empty subsets of a relation scheme $R$ is a partition of $R$ if the elements in the set are pairwise disjoint and the union of them is $R$. For any set $Y = \{Y_1, Y_2, \ldots, Y_i\}$ of subsets of $R$ such that $Y_1, Y_2, \ldots, Y_i = R$, there exists a unique partition $W = \{W_1, W_2, \ldots, W_j\}$ of $R$ such that (1) for any $Y_k$, $1 \leq k \leq i$, $Y_k$ is the union of some elements in $W$, and (2) if $Z$ is a partition of $R$ and any $Y_k$, $1 \leq k \leq i$, is the union of some elements in $Z$, then the cardinality of $Z$ is no less than that of $W$. $W$ is called the basis of $Y$.

By the inference rules $MO$ and $MS$, we see that for each non-empty $X_o \subseteq R$, there exists a set of multivalued dependencies.

$$X_o \rightarrow W_i, \quad W_i \neq \emptyset, \quad i = 1, 2, \ldots, k,$$
such that \(\{W_1, W_2, \ldots, W_k\}\) is the basis of the set \(\{Y - X_0; \ X_0 \rightarrow Y\ is\ in\ M^+,\ Y - X_0\ is\ non-empty\}\). We call the set \(\{W_1, W_2, \ldots, W_k\}\) the dependency basis of \(X_0\) with respect to \(M\) and denote it by \(Dep_M(X_0)\).

At the first instance, it may seem that the concepts of \(X_F^+\) of a set \(X\) with respect to a set \(F\) of functional dependencies and the concept of \(Dep_M(X)\) of a set \(X\) with respect to a set \(M\) of multivalued dependencies are different, as the former is a set of attributes, while the latter is a collection of sets of attributes. However, if we think of the collection of the singleton sets, \(\{(A): A \in X_F^+\}\). Then it is just the basis of the collection of sets that are functionally dependent on \(X\) by a functional dependency in \(F^+\). That this basis consists only of singleton sets follows from the decomposition rule for functional dependency. Thus, \(X_F^+\) and \(Dep_M(X)\) are indeed parallel concepts; the simpler representation of \(X_F^+\) is made possible by the stronger form of decomposition rule that applies to functional dependencies. Note that, in contrast to the concept of \(Dep_M(X)\) in Beeri [12], the \(Dep_M(X)\), as defined above, does not include subsets of \(X\) as its elements, that is, here \(Dep_M(X)\) is a partition of \(R - X\); we sometimes say that \(Dep_M(X)\) covers \(R - X\).

The similar approach, as used to solve the membership problem for functional dependencies, is used to solve the membership problem for multivalued dependencies. To decide if the multivalued dependency \(X \rightarrow Y\) is in \(M^+\), first compute \(Dep_M(X)\). Then \(X \rightarrow Y\) is in \(M^+\) if and only if \(Y - X\) is empty, or \(Y - X\) is a union of some set in \(Dep_M(X)\).
The following algorithm from Beeri [12] with obvious modifications computes the dependency basis of a subset $X$ of $R$ with respect to a set $M$ of multivalued dependencies.

### 2.2.4 Algorithm

```plaintext
procedure DEP-BASIS (X,M)
begin
    BASIS ← \{R-X\}
    change-flag ← T
    while change-flag do
        begin
            change-flag ← F
            for each $W \rightarrow Z$ in $M$ do
            begin
                $Y ← \phi$
                for each $W_i$ in BASIS do
                if $W_i \cap W ≠ \phi$ then $Y ← Y \cup W_i$
                $Z' ← Z/Y$
                if $Z' ≠ \phi$ then
                begin
                    for each $W_i$ in BASIS do
                    begin
                        $W' ← \phi$
                        if $W_i \subseteq Z'$ then $W' ← W' \cup W_i$
                    end
                end
            end
            if $Z' ≠ W'$ then do
            begin
                change-flag ← T
            end
        end
    end
end
```

62
We now analyze the time complexity of the above algorithm. Let \( m \) be the number of attributes in \( R \), and let \( m_1 \) be the number of dependencies in \( M \). We represent the sets of attributes as bit vectors of length \( m \). Then an operation on sets such as comparing two sets, or computing the difference, or union of sets requires time \( O(m) \). Since, at any time, \( BASIS \) can contain at most \( m \) elements (Recall that \( X=\emptyset \) is
possible), the innermost do loop can be executed in time $O(m^2)$. Similarly, computing $Z'$ also requires time $O(m^2)$. Thus, it may require time $O(m_1 m^2)$ in one loop until a change of value of BASIS is found. Therefore, the total time spent, in the worst case, is $O(m_1 m^3)$. The correctness of the above algorithm is proved by showing that $\text{Dep}_H(X)$ contains and is contained in the final value of BASIS returned at the end of the execution of the algorithm. At this point the membership algorithm is immediate, and it also runs in time $O(l(m)^4)$, where $l(m)$ is the size of the description of given set of dependencies [12]. It was later refined by Hagihara et al. [50] to an $O(\min(k^2 |U|, l(M)^2))$ algorithm, where $U$ is the set of attributes in $M$, $|U|$ is the size of $U$, and $k$ is the number of dependencies in $M$. Sagiv [86] used a somewhat different approach and solved the membership problem for multivalued dependencies in $O(l(M).|Y|)$, where $Y$ is the right side of the multivalued dependency whose membership in $H^+$ is to be checked. He also used this solution to derive an $O(l(M).p)$ algorithm for computing the dependency basis of a set $X$, where $p$ is the numbers of sets in the dependency basis of $X$. Galil [38] gave an $O((1+\min(k, \log p))l(M))$ algorithm for computing the dependency basis of a set of attributes $X$, where $k$ is the number of multivalued dependencies in $M$. Its worst case is $O(l(M) \log |U|)$ and is superior to the previous algorithms.

When both functional and multivalued dependencies are considered in combination, the membership problem can be stated as follows: Let $F$ be a set of functional dependencies and let $M$ be a set of multivalued dependencies. The
membership problem is to decide for any functional dependency or any multivalued dependency, whether it is in \((FUH)^+\).

Let us define for any set \(X\) the closure of \(X\) relative to \(FUH\), denoted by \(X_{FUH}^+\), to be the set of all attributes that are functionally dependent on \(X\) by a functional dependency in \((FUH)^+\). We also define the dependency basis of \(X\) with respect to \(FUH\), denoted by \(Dep_{FUH}(X)\), to be the basis of the collection of subsets of \(R-X\) that are dependent on \(X\) by a multivalued dependency in \((FUH)^+\). Note that if \(X+A\) is in \((FUH)^+\), then, by rule FM1, \(X\rightarrow A\) is also in \((FUH)^+\). Thus, for each attribute in \(X_{FUH}^+\), there is a singleton set in \(Dep_{FUH}(X)\), containing that attribute.

We note that if \(M\) is empty set, that is, only functional dependencies are given, then \(X_{FUH}^+\) is the set of attributes that are functionally dependent on \(X\) by a functional dependency that is derivable from by the rules F1-F3 only [14]. Thus, the definition of \(X_{FUH}^+\) is an extension of the definition of \(X_F^+\). Similarly, if \(F\) is empty, that is, only multivalued dependencies are given, then \(Dep_{FUH}(X)\) is the basis of the collection of subsets of \(R-X\) that depend on \(X\) by a multivalued dependency that is derivable from \(M\) by the rules MO-M3 only [14]. Thus, the definition of \(Dep_{FUH}(X)\) here is an extension of the definition of \(Dep_{M}(X)\).

Now, to decide if a given functional dependency \(X\rightarrow Y\) is in \((FUH)^+\), we need to be able to compute \(X_{FUH}^+\) so that we can check if \(Y \subseteq X_{FUH}^+\). To decide if a multivalued dependency \(X\rightarrow Y\) is in \((FUH)^+\), we need to be able to compute \(Dep_{FUH}(X)\) so that we can check if \(Y-X\) is a union of sets from \(Dep_{FUH}(X)\).
In order to compute $\text{Dep}_{\overline{\text{FUM}}}(X)$, functional dependencies in $F$ are reduced to multivalued dependencies using the rule $\text{FM1}$ [12]. Note that a functional dependency $X \rightarrow A_1A_2\ldots A_k$ in $F$ is considered to be the equivalent representation of functional dependencies $X \rightarrow A_1, X \rightarrow A_2, \ldots, X \rightarrow A_n$, that is, functional dependencies in $F$ are assumed to be of the form $X \rightarrow A$ where $A$ is a single attribute not in $X$. Beeri [12] has shown that $\text{Dep}_{\overline{\text{FUM}}}(X) = \text{Dep}_{\overline{\text{FUM}}}(X)$, where $\overline{F} = \{X \rightarrow Y : X \rightarrow Y \in F\}$. Since $\overline{\text{FUM}}$ contains only multivalued dependencies, we can compute $\text{Dep}_{\overline{\text{FUM}}}(X)$ using Algorithm 2.2.4, where the input in this case will be set of attributes $X$ and the set $\overline{\text{FUM}}$. Thus, the membership problem of a multivalued dependency in $(\overline{\text{FUM}})^+$ is solved. To decide if a given functional dependency $X \rightarrow Y$ is in $(\overline{\text{FUM}})^+$, $X^+_{\overline{\text{FUM}}}$ is computed by the following algorithm [12].

2.2.5 Algorithm

procedure CLOSURE $(X,F,M)$

begin

\text{BASIS} \leftarrow \text{DEP-BASIS} (X, \overline{\text{FUM}}) //use algorithm DEP-BASIS//

CLOSURE \leftarrow X

for each $A \in \mathcal{R} - X$ do

begin

if $\{A\} \in \text{BASIS}$ then

begin

for each $W \rightarrow B$ in $F$ do

if $B = A$ then CLOSURE \leftarrow CLOSURE \cup A

end

end

end
2.2.6 Theorem [12] : The algorithm described above always terminates. After it has terminated, the value of CLOSURE equals $X^+_{FUM}$. The time complexity of the algorithm is $O(l(FUM)^4)$, where $l(FUM)$ is the size of the description of the given dependencies.

Most algorithms in relational database theory use a set of dependencies as their input. The efficiency of these algorithms depend on the number of dependencies in the input set as well as on the number of attributes (size) involved in dependencies. It is precisely for this purpose, the equivalent representations of dependencies are investigated by many authors [44,66,72,76,96]. Starting point of such considerations is to formalize the notion of equivalence of two sets of dependencies. As our primary concern here are functional and multivalued dependencies, we, first, formalize the notions of equivalence of two sets of functional dependencies.

Let $F_1$ and $F_2$ be two sets of functional dependencies. Then $F_1$ and $F_2$ are said to be equivalent if $F_1^+ = F_2^+$. In that case we say that $F_1$ is a cover of $F_2$ and vice versa.

Once the membership problem for functional dependencies is solved, the problem of equivalence of two sets of functional dependencies can be reduced to the membership problem as follows: Two sets $F_1$ and $F_2$ are equivalent if and only if $f \in F_1$ implies $f \in F_2^+$, and $g \in F_2$ implies $g \in F_1^+$.

Given a set $F_1$ of functional dependencies, its cover set $F_2$ can be optimum in respect of several criteria. We say that $F_2$ is non-redundant cover of $F_1$ if no proper subset of it is a
cover of $F_1$. $F_2$ is called a minimum cover of $F_1$ if it has as few functional dependencies as any other cover of $F_1$. Our concern in both the above covers is the number of functional dependencies involved. Mair [68] defined the notion of optimum cover for a set of functional dependencies as follows: A cover $F_2$ of a set of functional dependencies $F_1$ is optimal if $\|F_2\| \leq \|F_1\|$, where $\|F_1\|$ and $\|F_2\|$ denote the number of attributes appearing in $F_1$ and $F_2$, respectively (Repeated symbols are counted as many times as they occur).

The yes/no minimum cover problem is: Given a set of functional dependencies $F$ and an integer $k$, is there a cover $F_1$ of $F$ with no more than $k$ functional dependencies. It is known that the above problem is NP-complete. What has actually been proved is that the yes/no contained cover problem is NP-complete. The contained cover problem is the minimum cover problem with added restriction that $F_1$ is contained in $F$. The optimal cover problem is the same as the minimum cover problem except that optimal cover must have fewer than $k$ attribute symbols (rather than functional dependencies); and it turns out that optimal cover problem is NP-complete [68].

The synthesis algorithm [19] is a prime example of why we are interested in finding small cover for a given set of dependencies: using a minimum cover minimizes the number of relations that need to be stored and an optimum cover minimizes the amount of storage required for storing the relations in third normal form (at least if the domains of the attributes are not considered).
Similarly, as in the case of functional dependencies, two sets of multivalued dependencies $M_1$ and $M_2$ are said to be equivalent if $M_1^+ = M_2^+$. In that case $M_1$ is called a cover of $M_2$ and vice versa $M_2$ is called a cover of $M_1$. There are different notions of minimality of covers for sets of multivalued dependencies. One common definition of a minimal cover of a set of multivalued dependencies is defined as follows. A set $M_2$ of multivalued dependencies is a minimal cover of another set $M_1$ of multivalued dependencies if $M_2$ is a cover of $M_1$ and no proper subset of $M_2$ is a cover of $M_1$. However, Ozsoyoglu and Yuan [76] noted that this definition is not sufficient to distinguish some redundancies in sets of multivalued dependencies; for instance, a minimal cover may have dependencies which contain redundant attributes, and also it may not be conflict-free. In an attempt to eliminate redundancies in a given set of multivalued dependencies, the concept of a reduced multivalued dependency was introduced as follows [76]:

2.2.7 Definition: Let $M$ be a set of multivalued dependencies in $R$. A multivalued dependency $X \rightarrow W$ is $M^+$ is said to be

(i) trivial, if $XW = R$, or $W \subseteq X$;

(ii) left reducible, if there is an $X' < X$ such that $X' \rightarrow W$ is in $M^+$;

(iii) right reducible, if there is a $W' < W$ such that $X \rightarrow W'$ is a nontrivial multivalued dependency in $M^+$;

(iv) transferable, if there is an $X' < X$ such that $X' \rightarrow (X-X')W$ is in $M^+$. 
A multivalued dependency $X \rightarrow \rightarrow W$ is said to be reduced if it is nontrivial, left-reduced (non-left-reducible), right-reduced (non-right-reducible), and non-transferable.

2.2.8 Definition: A set $M$ of multivalued dependencies is said to be minimal if (i) each multivalued dependency in $M$ is reduced, and (ii) no proper subset of $M$ is a cover of $M$.

The difference between this notion of minimal cover and the usual one as defined earlier in this section is thus the fact that a multivalued dependency is reduced in the former (cover in Definition 2.2.8); but may not be reduced in the latter.

2.3 Functional and Multivalued Dependencies and Nulls

In Chapter 1 we have seen that the problem of determining functional dependencies following Janas [54-58] approach requires the introduction of null values in the domains of the attributes. Subsequently we argued that Zaniolo's [110] "no-information" interpretation of null value is the most primitive and with well understood semantics. The treatment of functional and multivalued dependencies in Lein [62] (see also Atzeni and Morfuni [10]) is suitable of applications to null values under the no-information interpretation. In order to introduce the notions of functional and multivalued dependencies when nulls with "no-information" interpretations are present in the domains of the attributes, we introduce some preliminaries.

A null value, denoted by the symbol "⊥", and interpreted as "no-information" is introduced in the domains of attributes of the universe under consideration. Thus, the
domain of an attribute \( A \) now becomes \( \text{dom } A \cup \{ \bot \} \). In the definition of a relation \( r \) over a relation scheme \( R \), introduced in Section 1.1, the domain of each attribute \( A \) in \( R \) is replaced by \( \text{dom } A \cup \{ \bot \} \). The restriction of a tuple \( t \) in \( r \) on an attribute \( A \) in \( R \), that is \( t[A] \), is either in \( \text{dom } A \) or \( t[A]=\bot \). A tuple \( t \) in \( r \) is \( X \)-total \((X \subseteq R)\) if \( t[A] \neq \bot \) for each \( A \) in \( X \). Let \( t_1 \) and \( t_2 \) be two tuples in \( r \). \( t_1 \) subsumes \( t_2 \) if either \( t_1[A]=t_2[A] \) or \( t_2[A]=\bot \) for each \( A \) in \( R \); equivalently we say that \( t_1 \) is more informative than \( t_2 \). If \( t_1 \) subsumes \( t_2 \), and vice versa, then \( t_1 \) and \( t_2 \) are identical. The relations considered in this thesis are subsumption free. The notion of being more informative can be extended to relations, which will said to be more informative than or to subsume other relations.

A relation \( r_1 \) subsumes a relation \( r_2 \), when for each tuple \( t_2 \in r_2 \) there is a tuple \( t_1 \in r_1 \) such that \( t_1 \) subsumes \( t_2 \). The relations \( r_1 \) and \( r_2 \) are information-wise equivalent when \( r_1 \) subsumes \( r_2 \) and vice versa \( r_2 \) subsumes \( r_1 \).

Let \( r \) be a relation over \( R \), and \( X \) and \( Y \) are non-empty subsets of \( R \). An operation subsume, which when applied to a relation removes from it all tuples that are subsumed by other tuple in the relation, can be defined. The projection of \( r \) on \( X \) is the relation

\[
\text{subsume } \{ t[X]: t \text{ is in } r \}
\]

The \( X \)-total projection of \( r \) on \( Y \), written \( r_X[Y] \), is the relation

\[
\text{subsume } \{ t[Y]: t \text{ is in } r, \text{ and } t \text{ is } X\text{-total} \}.
\]
Let \( t' \) be a tuple in a relation \( r' \) over a relation scheme \( Y \subseteq R \).

We use \( r_Y = t', [X] \) to denote the relation.

\[
\text{subsume} \{ t[X]: t \text{ is } r \text{ and } t[Y] = t' \}.
\]

Let \( r_1, r_2, \ldots, r_n \) be relations over a relation scheme \( R \).

Then the union of \( r_1, r_2, \ldots, r_n \) is the relation

\[
\bigcup_{i=1}^{n} r_i = \text{subsume} \{ t: t \text{ is a tuple in some } r_i, i=1,2, \ldots, n \}.
\]

### 2.3.1 Functional Dependencies with Nulls

According to Lien [62], a functional dependency with nulls (NFD) \( X \rightarrow Y \) holds in a relation \( r \) over a relation scheme \( R \) (with \( XY \subseteq R \)) when, for each pair of \( X \)-total tuples \( t_1 \) and \( t_2 \) in \( r \), if \( t_1[X] = t_2[X] \), then \( t_1[Y] = t_2[Y] \).

For null-free relations (that is classical relations) the definition of NFD reduces to that of functional dependency and so it is a correct generalization of the concept. Moreover, it is coherent with the no-information interpretation. In fact, tuples with nulls in attributes in \( X \) cannot cause a violation of a dependency \( X \rightarrow Y \): the nulls mean that no-information is available about those attributes. On the other hand, two \( X \)-total tuples, \( t_1, t_2 \) such that \( t_1[X] = t_2[X] \), and \( t_2 \) is \( A \)-total while \( t_1 \) is not, violate a dependency \( X \rightarrow Y \) with \( A \in Y \): the tuple \( t_1 \) indicates that no-information is available about the value for \( A \) associated with \( t_1[X] \), while the tuple \( t_2 \) indicates that the value for \( A \) associated with \( t_1[X] = t_2[X] \) does exist, and this violates the natural definition of functional dependency that if the values for \( X \) are the same for two tuples, both tuples must contain the same information for the attributes in
Y. It should be noted that this definition of satisfaction refers to properties of our knowledge of the real world, while it is generally stated that integrity constraints express properties of the real world itself. On the other hand, for databases without nulls it is simply assumed that the real world is represented faithfully, and so that our knowledge is complete, and coincide with the real world; but, even in this case, databases are approximations of the real world, and our knowledge is far from being complete and dependencies are compared with the available data, that is, our knowledge.

In the NFD $X \rightarrow Y$, $X$ and $Y$ are said to left and right side, respectively.

With respect to the inference rules it is immediate to prove that reflexivity augmentation, union, and decomposition are sound rules for NFDs also, while transitivity is not as shown by the counter example relation in Figure 1 below, which satisfies

\begin{verbatim}
    A  B  C
    ---
    a_1  c_1
    a_1  c_2

Fig.1
\end{verbatim}

both $A \rightarrow B$ and $B \rightarrow C$ but does not satisfy $A \rightarrow C$. It is clear from the example that the unsoundness of the rule is caused by the presence of nulls in the attributes $Y$ ($B$ in the example) which implements the transitivity.

However, the other four rules are complete for the derivation of NFDs [82]. In order to state the theorem to this
effect, we introduce the following definitions. In the sequel
F will denote a set of NFDs.

A functional dependency with nulls $X \rightarrow Y$ can be derived
from F, written $F \vdash X \rightarrow Y \in F$ if $X \rightarrow Y \in F$, or $X \rightarrow Y$ can be obtained
from F by a finite number of the application of the inference
rules $F_1$, $F_2$, $F_5$, and $F_6$.

A functional dependency with nulls $X \rightarrow Y$ is implied by F if
in every relation in which all the dependencies of F hold, $X \rightarrow Y$
also holds. By $F^+$, we denote the set of all functional
dependencies with nulls implied by F.

2.3.1.1 Theorem: Let F be a set of NFDs. Then

$$X \rightarrow Y \in F^+ \iff F \vdash X \rightarrow Y.$$  

As for the classical functional dependencies, the concept
of closure of a set of attributes with respect to a set of
NFDs can be defined and used as the basis for the membership
algorithm. We indicate the closure of X with respect to a set
F of NFDs with $X^+_F$. On the other hand, the closure cannot be
computed by means of algorithm 2.2.3, because the transitivity
rule is not sound, and the attributes added to the initial
value of the variable CLOSURE cannot be used to add further
attributes. The algorithm can be modified by replacing the
current value of the variable CLOSURE in the comparison in
the if statement with its initial value. As a consequence
each NFD can be used at most once to add attributes to CLOSURE,
and so the external loop can be eliminated. The following
algorithm, which computes $X^+_F$ of a set X with respect to a set
F of NFDs, is from Atzeni and Morfuni [11].
2.3.1.2 Algorithm

procedure CLOSURE(X,F)
begin
    CLOSURE ← X
    for each V→W in F do
        if V≤X then CLOSURE ← CLOSURE ∪ W
    end

If X≤R, |R|=m, and |F|=m₁, then it is easily seen that the run time of the above algorithm is O(m₁m). Atzeni and Morfuni [11] proved that the final value of the variable CLOSURE returned by the above algorithm equals X⁺_F; For the sake of completeness, we shall write the proof of this result here.

2.3.1.3 Theorem: Algorithm 2.3.1.2 correctly computes the closure X⁺_F.
Proof: We show that the final value of the variable CLOSURE, indicated by CLOSURE*, is equal to X⁺_F.

(i) CLOSURE* ≤ X⁺_F. Let A ∈ CLOSURE*; if A ∈ X, A is trivially in X⁺_F; otherwise, there is an NFD V→W in F such that V≤X and A∈W; in this case, X→A can be obtained from F using the augmentation and decomposition rules, and so A is in X⁺_F.

(ii) X⁺_F ≤ CLOSURE*. Let A be in X⁺_F: this means that X→A is in F⁺ and so (since the rules are complete) is derivable from F by means of the rules F1, F2, F5, and F6. We prove the theorem by induction on the length of the derivation, with the following inductive hypothesis: if Z≤X and Z→Y is derivable in not more than s steps, then Y is contained in CLOSURE* (we use
a possibly non-singleton set \( Y \) in the proof because intermediate NFDs in the derivation need not have singletons as right-hand-sides).

**Basis:** \( s=1 \). \( Z \rightarrow Y \) is in \( F \) and so, when \( Z \rightarrow Y \) is processed by the algorithm (and this will happen) \( Y \) is added to CLOSURE.

**Induction:** \( s>1 \) and the inductive hypothesis holds for derivations of length less than \( s \). \( Z \rightarrow Y \) is the last NFD in the derivation, and it is there because it is in \( F \) (and in this case we can argue as above), or because it is derived from other NFDs by means of an inference rule. So, we have four cases, one for each rule.

1. **Reflexivity:** If \( Y \subseteq Z \subseteq X \), it is included in CLOSURE since the beginning.

2. **Augmentation:** There are an NFD \( V \rightarrow W \), derived in less than \( s \) steps, and a set of attributes \( T \) such that \( VT=Z \) and \( WT=Y \); since \( V \subseteq Z \subseteq X \), \( V \rightarrow W \) is derived in less than \( s \) steps, \( W \) is in CLOSURE* and since \( T \subseteq X \) (and so it is in CLOSURE since the beginning), \( Y \subseteq \text{CLOSURE}^* \).

3. **Union:** There are two NFDs \( Z \rightarrow V \) and \( Z \rightarrow W \), both derivable in less than \( s \) steps, such that \( VW=Y \); by the inductive hypothesis, since \( Z \subseteq X \), both \( V \) and \( W \) are included in CLOSURE* and so is \( VW \).

4. **Decomposition:** There is an NFD \( Z \rightarrow W \), derivable in less than \( s \) steps such that \( Y \subseteq W \); by the inductive hypothesis, since \( Z \subseteq X \), \( W \subseteq \text{CLOSURE}^* \) and so (since \( Y \subseteq W \)) \( Y \subseteq \text{CLOSURE}^* \).

The membership of an NFD \( X \rightarrow Y \) in the closure \( F^+ \) of a set \( F \) of NFDs is immediate in view of the following lemma.
2.3.1.4 Lemma: Let \( F \) be set of NFDs. Then \( F \upharpoonright X \rightarrow Y \) if and only if \( Y \subseteq X_F^+ \).

Proof: Let \( F \upharpoonright X \rightarrow Y \). Let \( Y = A_1 A_2 \ldots A_k \). Then \( X \rightarrow A_i \), \( i = 1, 2, \ldots, k \), can be derived using \( F \). Therefore, \( X \rightarrow A_i \), \( i = 1, 2, \ldots, k \), is in \( F^+ \), and so \( A_i \), \( i = 1, 2, \ldots, k \), is in \( X_F^+ \). Thus, \( Y \subseteq X_F^+ \).

Conversely, let \( Y \subseteq X_F^+ \), where \( Y = A_1 A_2 \ldots A_k \). Since \( Y \cap X \subseteq X \), \( X \rightarrow Y \cap X \) can be derived using \( F \). Also, for each attribute \( A_i \in X_F^+ - X \), \( A_i \in Y \), there exists an NFD \( V \rightarrow W \) in \( F \) such that \( V \subseteq X \) and \( A_i \subseteq W \). From \( V \rightarrow W \), \( X \rightarrow W \) can be derived using \( F \). Taking the union of right sides of all such NFDs \( X \rightarrow W \), and denoting this union by \( Y_1 \), we can derive the NFD \( X \rightarrow Y_1 \) using \( F \). Since \( Y \cap X \subseteq X_F^+ - X \subseteq Y_1 \), we have \( X \rightarrow Y - X \) using \( F \). Now \( X \rightarrow Y \) can be derived using \( F \).

2.3.2 Multivalued Dependencies with Nulls

According to Lien [62], a multivalued dependency with nulls (NMVD) \( X \leftrightarrow Y \) holds in a relation \( r \) over a relation scheme \( R \) (with \( XY \subseteq R \)) if for every tuple \( t \) in \( r_X[XZ] \), where \( Z = R - XY \), it is true that

\[
\text{r}_X[XZ] = t[Y] = t[X][Y].
\]

It is important to note that NMVDs are defined in the context of the entire relation scheme \( R \). Whether \( X \leftrightarrow Y \) holds in a relation \( r \) over a relation scheme \( R \) or not depends also on the complement \( Z \) of \( XY \) in \( R \). Again, \( X \) and \( Y \) are called left and right sides of \( X \leftrightarrow Y \), respectively.

For null-free relations the definition of NMVD reduces to that of multivalued dependency.

The following theorem characterizes equivalently the notion of an NMVD [62].
2.3.2.1 Theorem: An NMVD $X \leftrightarrow Y$ holds in a relation $r$ over a relation scheme $R$ if and only if for all $X$-total tuples $t_1$ and $t_2$ in $r$ such that $t_1[X] = t_2[X]$, there exist tuples $t_3$ and $t_4$ in $r$ such that

$t_3[XY] = t_1[XY], t_3[R-XY] = t_2[R-XY]$ and

$t_4[XY] = t_2[XY], t_4[R-XY] = t_1[R-XY]$.

In the sequel, $M$, possibly with subscripts, denotes a set of NMVDs—even if it is not mentioned explicitly. Lein has shown that the inference rules $M_0, M_1, M_2, M_5$, and $M_8$ are sound for NMVDs. However, first four of these inference rules are sufficient for proving completeness. Next, we shall formally state this fact and quote the proof of completeness of these inference rules for NMVDs from [62]. However, before that can be done, we need the following definitions.

As for classical multivalued dependencies, the concept of dependency basis of a set of attributes with respect to a set of NMVDs can be defined and used. The dependency basis of a set of attributes $X$ with respect to a set $M$ of NMVDs is still denoted by $\text{Dep}_M(X)$ and is defined as the basis of the set $\{Y-X: X \leftrightarrow Y \in M^+ \text{ and } Y-X \text{ is non-empty}\}$.

A multivalued dependency with nulls $X \leftrightarrow Y$ can be derived from $M$, written $M \models X \leftrightarrow Y$ if $X \leftrightarrow Y \in M$ or $X \leftrightarrow Y$ can be obtained from $M$ using the inference rules $M_0, M_1, M_2$ and $M_5$.

A multivalued dependency with nulls $X \leftrightarrow Y$ is defined by $M$, if every relation in which all the NMVDs of $M$ hold, $X \leftrightarrow Y$ holds. By $M^+$, we denote the set of all NMVDs implied by $M$. 

78
2.3.2.2 Theorem: Let \( M \) be a set of NMVDs. Then
\[
M \models X \leftrightarrow Y \iff X \leftrightarrow Y \in M^+.
\]

Proof: The implication (\( \models \)) is, in fact, the soundness of the inference rules for NMVDs.
(\( \iff \)). In order to prove this implication, we need only establish the following claim.

Claim: Let \( M \) be set of NMVDs on a relation scheme \( R \) such that if \( \phi \leftrightarrow V \) is in \( M \) then \( V=R \). For any NMVD \( g \) on \( r \) that is not in \( M^+ \) there exists a relation \( r \) over \( R \) such that every NMVD in \( M^+ \) holds in \( r \) but \( g \) does not.

Proof of Claim: Suppose \( g \) is \( X \leftrightarrow V \), known to be not in \( M^+ \). Let
\[
\text{Dep}_M(X) = \{W_1, W_2, \ldots, W_m\}.
\]
We shall construct such a relation \( r \) over \( R \) with only data values from \((a_1, a_2, \ldots, a_m, 0, 1)\) and, of course, \( \bot \). The relation contains \( 2m \) tuples \( \{t_1, t_2, \ldots, t_{2m}\} \).

The definition of these tuples are, for \( i=1, 2, \ldots, m \),
(a) for any attribute \( A \) in \( X \), \( t_{2i-1}[A] = t_{2i}[A] = a_i \);
(b) for any attribute \( A \) in \( W_i \), \( t_{2i-1}[A] = 0 \) and \( t_{2i}[A] = 1 \);
(c) for any attribute \( A \) in \( W_j \), such that \( i \neq j \), \( t_{2i-1}[A] = t_{2i}[A] = \bot \)

Next we shall verify that for any NMVD \( Y \leftrightarrow Z \) in \( M^+ \), \( Y \leftrightarrow Z \) holds in \( r \). The proof is divided into several cases.

Case 1: \( Y \cap W_i \neq \phi \) and \( Y \cap W_j \neq \phi \) for \( i \neq j \). Then \( Y \leftrightarrow Z \) holds trivially in \( r \). This is because every tuple is not \( Y \)-total.

Case 2: There is one \( i \) such that \( Y \subseteq X \cup W_i \). First we notice that \( Y \leftrightarrow Z \cap X \) holds in \( r \). Next we prove \( Y \leftrightarrow Z \cap W_j \) holds in \( r \) for any \( j \) by cases:
(a) \( Z \cap W_j = \phi \), \( Y \leftrightarrow \phi \) holds in \( R \) by reflexivity.
(b) \( Z \cap W_j = W_j \). Either \( Y \) is a subset of \( X \) or \( Y \cap W_i \neq \phi \). For the
former case, $Y \rightarrow W_j$ is easily verified. For the latter, the only $Y$-total tuples are $t_{2i-1}$ and $t_{2i}$. Agian $Y \rightarrow W_j$ holds in $r$.

(c) $Z \cap W_j \neq \emptyset$ and $Z \cap W_j \neq W_j$. In this case $Y \cap W_i$ must be non-empty. For otherwise, $Y \subseteq X$ and $X \rightarrow Z$ by the augmentation rule. It then follows that $X \rightarrow Z \cap W_j$. This contradicts the assumption that $W_j$ is in the dependency basis. Since $Y \cap W_i$ is non-empty, only $t_{2i-1}$ and $t_{2i}$ are $Y$-total. So $Y \rightarrow Z \cap W_j$ can be easily verified.

Therefore, $Y \rightarrow Z \cap X$ and $Y \rightarrow Z \cap W_j$ for all $j$ hold in $r$. By union rule $(M5)$, $Y \rightarrow Z$ holds in $r$.

Finally, we need to show that $g : X \rightarrow V$ does not hold in $r$. There must be a $j$ such that $W_j \cap V \neq \emptyset$ and $W_j \cap V \neq W_j$ otherwise, $V$ is the union of some members of the dependency basis, and $g$ is in $M^+$. Consider $X \rightarrow V \cap W_j$

$$r_X = a_j^{[V \cap W_j]} = \{(0,0,...,0),(1,1,...,1)\}$$

and

$$r_X = a_j, W_j - V = (0,0,...,0), R - X - W_j = \emptyset [V \cap W_j] = \{(0,0,...,0)\}$$

Therefore, $X \rightarrow V \cap W_j$ does not hold. Since $X \rightarrow W_j$ does hold in $r$, $X \rightarrow V$ must not hold in $r$. This completes the proof of the theorem.

It should be noted that in the proof of completeness of the above theorem, no NMVD has an empty set on the left side. From our further consideration, we exclude NFDs and NMVDs with empty left sides.

After establishing the completeness of a formal system for NMVDs, our next concern is the membership problem: Given an NMVD $X \rightarrow Y$ and a set of NMVDs $M$, is $X \rightarrow Y \in M^+$? In order to solve this problem, we must be able to compute $\text{Dep}_M(X)$; so now
we present an algorithm to compute $\text{Dep}_M(X)$, which, in fact, is a modification of Algorithm 2.2.4.

2.3.2.3 Algorithm:

procedure DEP-BASIS($X, M$)

begin

    BASIS $\leftarrow \{R-X\}$

    for each $W \rightarrow Z$ in $M$ do

        if $W \subseteq X$ then

            begin

                ABASIS $\leftarrow$ BASIS

                for each $W$ in BASIS do

                    begin

                        $W' \leftarrow \emptyset$

                        $W'' \leftarrow \emptyset$

                        if $W_i \cap Z \subseteq \emptyset$ then

                            begin

                                if $W_i \neq Z$ then

                                    begin

                                        $W' \leftarrow W_i \cap Z$

                                        $W'' \leftarrow W_i / Z$

                                        ABASIS $\leftarrow \{W'\} \cup \{W''\} \cup \text{ABASIS}/W_i$

                                    end

                                end

                            end

                        end

                    end

                end

            end

        end

    BASIS $\leftarrow$ ABASIS

end

end
In the above algorithm the time spent in calculating the new value of BASIS is \(O(m^2)\), where \(m\) is the number of attributes in \(R\). Therefore, the total time spent in calculating the final value of BASIS is \(O(m_1 m^2)\), where \(m_1\) is the cardinality of \(M\).

2.3.2.4 Theorem: The algorithm described above computes correctly the value of \(\operatorname{Dep}_M(X)\), where \(\phi \neq X \subseteq R\).

Proof. Let \(\overline{\operatorname{Dep}}_M(X)\) be the value of BASIS after the algorithm is terminated. We shall prove that \(\overline{\operatorname{Dep}}_M(X) = \operatorname{Dep}_M(X)\). Let \(\overline{\operatorname{Dep}}_M(X) = \{Y_1, Y_2, \ldots, Y_k\}\). We first show that for each \(Y_i \in \operatorname{Dep}_M(X)\), \(X \leftrightarrow Y_i\) is in \(M^+\).

By reflexivity rule, \(X \leftrightarrow A \in M^+\) for each \(A \in X\) and so is \(X \leftrightarrow X\) by union rule. By complementation rule \(X \leftrightarrow R-X \in M^+\). Thus, for the set \(R-X\) which is the only element in the initial value of BASIS, \(X \leftrightarrow R-X\) is in \(M^+\). We show, by induction, on the number of passes through the for loop that this is true after each pass through that loop. Suppose that the claim is true after \(j\), \(j \geq 0\), passes through the loop. Let \(W \leftrightarrow Z\) be the MNVD used in the next pass and suppose that the value of BASIS is changed in the pass (otherwise there is nothing to prove). Then \(W\) has to be a subset of \(X\); hence \(W \leftrightarrow Z\) can be augmented to \(X \leftrightarrow Z\). Since, by the induction hypothesis, for each set \(Y_i\) in (the current value) BASIS, \(X \leftrightarrow Y_i\) is in \(M^+\), it follows, by decomposition rule, that the claim is true for the new value of BASIS. Since \(\overline{\operatorname{Dep}}_M(X)\) is the final value of BASIS, \(\overline{\operatorname{Dep}}_M(X) = \overline{\operatorname{Dep}}_M(X)\), therefore, for each set \(Y_i \in \overline{\operatorname{Dep}}_M(X)\), \(X \leftrightarrow Y_i\) is in \(M^+\).

Since, for each \(Y_i\), \(1 \leq i \leq k\), \(X \leftrightarrow Y_i \in M^+\), \(Y_i\) can be expressed as a union of the elements of \(\overline{\operatorname{Dep}}_M(X)\). To
conclude the proof that $\text{Dep}_M(X) = \overline{\text{Dep}_M(X)}$, we now show that each element of $\text{Dep}_M(X)$ is a union of elements of $\overline{\text{Dep}_M(X)}$. Since both collections are partitions of $R-X$, equality follows. We construct a relation $r$ with the following properties:

(a) Every NMVD in $M$ holds in $r$.

(b) An NMVD $X \leftrightarrow Y$ holds in $r$ if and only if either $Y-X=\emptyset$ or $Y-X$ is a union of elements of $\overline{\text{Dep}_M(X)}$.

Since every NMVD in $M$ holds in $r$, so is every NMVD in $M^\dagger$. Hence, for each $Y \in \text{Dep}_M(X)$, $X \leftrightarrow Y$ holds in $r$. Therefore, by (b), $Y$ is a union of the elements of $\overline{\text{Dep}_M(X)}$.

The relation $r$ is constructed with only the data values from $\{a_1, a_2, ..., a_k, 0, 1\}$ and, of course, $\perp$. The relation contains $2k$ tuples $(t_1, t_2, ..., t_{2k})$. The definitions of these tuples are for $i = 1, 2, ..., k$,

$$t_{2i-1}[A] = t_{2i}[A] = a_i \text{, if } A \in X,$$

$$t_{2i-1}[A] = 0 \text{ and } t_{2i}[A] = 1 \text{ if } A \in Y_i$$

and

$$t_{2i-1}[A] = t_{2i}[A] = \perp, \text{ otherwise}.$$

We now show that each of the NMVD in $M$ holds in $r$. Let $Y \leftrightarrow Z$ be any NMVD in $M$. The proof is divided in the following case of:

Case 1. $Y \cap Y_i \neq \emptyset$ and $Y \cap Y_j \neq \emptyset$ for $i \neq j$. Then $Y \leftrightarrow Z$ holds trivially in $r$. This is because every tuple is not $Y$-total.

Case 2. There is one $i$ such that $Y \subseteq X \cup Y_i$. First we notice that $Y \leftrightarrow Z \cap X$ holds in $r$. Next we prove that $Y \leftrightarrow Z \cap Y_j$ holds in $r$ for any $j$ by cases:
(a) \( Z \cap Y_j = \emptyset \). Then \( Y \rightarrow \emptyset \) holds in \( r \) by reflexivity.

(b) \( Z \cap Y_j = Y_j \). Then, either \( Y \) is a subset of \( X \) or \( Y \cap Y_j \neq \emptyset \).

For the former case, \( Y \rightarrow Y_j \) is easily verified.

For the latter, the only \( Y \)-total tuples are \( t_{2i-1} \) and \( t_{2i} \). Again \( Y \rightarrow Y_j \) holds in \( r \).

(c) \( Z \cap Y_j \neq \emptyset \) and \( Z \cap Y_j \neq Y_j \). In this case, \( Y \cap Y_j \neq \emptyset \), for otherwise \( Y \subseteq X \). Since \( Y \subseteq X \) and the algorithm has terminated \( Z - X \) should be expressible as a union of the elements in \( \text{Dep}_M(X) \). This is contradiction to the fact that \( Z \cap Y_j \neq \emptyset \) and \( Z \cap Y_j \neq Y_j \). Therefore \( Y \cap Y_j \neq \emptyset \). Since \( Y \cap Y_j \neq \emptyset \). The only \( Y \)-total tuples are \( t_{2i-1} \) and \( t_{2i} \).

So \( Y \rightarrow Z \cap Y_j \) can be easily verified.

Therefore, \( Y \rightarrow Z \cap X \) and \( Y \rightarrow Z \cap Y_j \) for all \( j \), \( 1 \leq j \leq k \), hold in \( r \).

By the union rule \( Y \rightarrow Z \) holds in \( r \).

Now suppose that \( X \rightarrow Y \) holds in \( r \) and \( Y - X \neq \emptyset \) (otherwise the proof is trivial). Therefore, for each \( Y_i \in \overline{\text{Dep}}_M(X) \) \( X \rightarrow Y \cap Y_i \) holds in \( r \). However, \( X \) does not intersect any \( Y_i \in \overline{\text{Dep}}_M(X) \), so \( X \rightarrow Y \cap Y_i \) holds in \( r \) if and only if \( Y \cap Y_i \) is either empty or all of \( Y_i \). Thus \( Y - X \) is a union of some of \( Y_i \)'s, \( Y_i \in \overline{\text{Dep}}_M(X) \). This completes the proof.

2.3.2.5 Theorem: The membership problem for NMVDs can be decided in time \( O(1(M)^3) \) where \( 1(M) \) is the size of the description of the given set \( M \) of NMVDs.

Our main purpose here is to give the algorithm to compute \( \overline{\text{Dep}}_M(X) \) parallel to the one given in Algorithm 2.2.4 and to prove its correctness. Lien [62] has shown that \( \overline{\text{Dep}}_M(X) \) can be computed from the essential keys which are subsets of \( X \).
2.3.3 Functional and Multivalued Dependencies with Nulls

In Section 2.3.1, we noted a formal system for a family of NFDs and in Section 2.3.2, for a family of NMVDs. When NFDs and NMVDs are considered together, the additional inference rule valid is FM1. In order to present a complete formal system for a family of NFDs and NMVDs, and at the same time to be coherent in our presentation, we define the following.

A functional dependency with nulls $X \rightarrow Y$ (resp. A multivalued dependency with nulls $X \leftrightarrow Y$) can be derived from $\text{FuM}$, written $\text{FuM} \vdash X \rightarrow Y$ (resp. $\text{FuM} \vdash X \leftrightarrow Y$), if $X \rightarrow Y \in F$ (resp. $X \leftrightarrow Y \in M$), or $X \rightarrow Y$ (resp. $X \leftrightarrow Y$) can be derived from $\text{FuM}$ using the inference $F1$, $F2$, $F5$ and $F6$ (resp. $F1$, $F2$, $F5$, $F6$, $M0$, $M1$, $M2$, $M5$ and $FM1$).

A functional dependency with nulls $X \rightarrow Y$ (resp. A multivalued dependency with nulls $X \leftrightarrow Y$) is implied by $\text{FuM}$ if in every relation in which all of the NFDs of $F$ and all of the NMVDs of $M$ hold, $X \rightarrow Y$ (resp. $X \leftrightarrow Y$) also holds.

The following theorem from Lien [52] gives the soundness and the completeness of the formal system for NFDs and NMVDs.

2.3.3.1 Theorem: Let $F$ be a set of NFDs and $M$ be a set of NMVDs.

Then

(a) $X \rightarrow Y \in (\text{FuM})^+ \iff \text{FuM} \vdash X \rightarrow Y$,

(b) $X \leftrightarrow Y \in (\text{FuM})^+ \iff \text{FuM} \vdash X \leftrightarrow Y$.

From the previous results we can conclude that NFDs and NMVDs are much simpler to handle than classical functional and multivalued dependencies. For one thing, they do not have
powerful inference such as transitivity or pseudotransitivity. Another significant implication of completeness results for NFDs and NMVDs is the clear separation of functional properties from multivalued properties in a given set of dependencies. For instance, closure of a set $X$ with respect to a set $F$ of NFDs and a set $M$ of NMVDs, denoted by $X^+_{FUM}$, is the set, 
\[ \{ A : X \rightarrow A \in (F \cup M)^+ \} . \]
It follows that $X \rightarrow A \in (F \cup M)^+ \iff FUM \mid X \rightarrow A \iff F \mid X \rightarrow A \iff X \rightarrow A \in F^+$. Therefore, $X^+_{FUM} = X^+_F$. Thus, the closure of a set $X$ with respect to $FUM$ can be computed from Algorithm 2.3.1.2 using NFDs in $F$ only. Consequently, the membership problem of an NFD $X \rightarrow Y$ in $(FUM)^+$ can be decided essentially by, first, computing $X^+_F$, and then by checking whether $Y$ is contained in $X^+_F$.

Similarly, NMVDs implied by $FUM$ can be derived from $FUM$ by treating all NFDs as NMVDs (that is, apply FM1) and using only the rules M0, M1, M2 and M5. The intricate interaction which has been observed between the classical functional and multivalued dependencies [12] is absent from NFDs and NMVDs.

As for functional and multivalued dependencies, the notion of dependency basis of a set of attributes can be defined and used for NFDs and NMVDs. Given a set $F$ of NFDs and a set $M$ of NMVDs, the dependency basis of a set of attributes $X$ with respect to $FUM$, denoted by $\text{Dep}_{FUM}(X)$, is the basis of the set \{ $Y \in X : X \rightarrow Y \in (FUM)^+$, $Y-X$ is non-empty \}. According to this definition, it should be noted that if $A$ be an attribute in $X^+_F - X$, then $X \rightarrow A \in (FUM)^+$; and so the attribute $A$ will appear as a singleton set $\{ A \}$ in $\text{Dep}_{FUM}(X)$. Thus, $\text{Dep}_{FUM}(X)$ is a collection of subsets of $R$ (where $R$ is an underlying relation.
scheme) such that each attribute \( A \) in \( X_F^+ - X \) appears in \( \text{Dep}_{\text{FuM}}(X) \) as the singleton set \( \{A\} \) and the union of other elements in \( \text{Dep}_{\text{FuM}}(X) \) (that is the elements which do not intersect with \( X_F^+ \)) is \( R-X_F^+ \). Most often in Chapter 5, we are interested only in those elements of \( \text{Dep}_{\text{FuM}}(X) \) which are contained in \( R-X_F^+ \). For this reason, we define the dependency basis of \( X \) with respect to \( \text{FuM} \) covering \( R-X_F^+ \) as the basis of the set

\[
\{Y-X_F^+: X \rightarrow Y \in (\text{FuM})^+ \text{ and } Y-X_F^+ \text{ is non-empty}\}.
\]

This dependency basis is denoted by \( \text{Dep}_{\text{FuM}}(X) \).

Next we give an algorithm to compute the dependency basis of a non-empty set \( X \subseteq R \) with respect to set \( F \) of NFDs and a set \( M \) of NMVDs, which covers \( R-X_F^+ \).

### 2.3.3.2 Algorithm

```
procedure DEP-BASIS(X+,F,M)
begin
  CLOSURE + CLOSURE(X,F) //use Algorithm 2.3.1.3//
  BASIS + \{R-CLOSURE\}
  for each \( U \rightarrow W \) in \( M \) do
  if \( U \subseteq X \) then
  begin
    ABASIS + BASIS
    for each \( W_i \) in BASIS do
    begin
      \( W' + \emptyset \)
      \( W'' + \emptyset \)
      if \( W_i \cap W \neq \emptyset \) then
  
```

87
begin
  if \( W_i \subseteq W \) then
    begin
      \( W' \leftarrow W_i \cap W \)
      \( W'' \leftarrow W_i / Z \)
      \( ABASIS \leftarrow \{W'\} \cup \{W''\} \cup ABASIS / W_i \)
    end
  end
end

BASIS \leftarrow ABASIS
end

The time spent in the above algorithm in calculating \( X_F^+ \) is \( O(m_1 m) \), where \( |R| = m \) and \( |F| = m_1 \). Therefore, the execution time of the above algorithm is \( O((m_1 + m_2)m^2) \), where \( |M| = m_2 \).

The fact that the final value of the variable BASIS returned by Algorithm 2.3.3.2 equals \( \text{DepFUM}(X_+) \) is proved in the following theorem.

2.3.3.3 Theorem: Algorithm 2.3.3.2 computes correctly the value of \( \text{DepFUM}(X_+) \), where \( \phi \neq X \subseteq R \).

Proof: Let \( \text{DepFUM}(X_+) \) be the value of BASIS after the algorithm has terminated. We shall prove that \( \text{DepFUM}(X_+) = \text{DepFUM}(X_+) \). First, we shall show that \( \text{DepFUM}(X_+) \) covers \( R-X_F^+ \). In view of Algorithm 2.3.1.2 the value of CLOSURE equals \( X_F^+ \). Since in the first for loop we begin with a partition of \( R-CLOSURE \) which is \( R-CLOSURE \) itself, it follows that the final value of BASIS is also a partition of \( R-X_F^+ \). Thus, \( \text{DepFUM}(X_+) \)
covers $R-X^+_F$. Let $\overline{\text{Dep}_{\text{FUM}}(X_+)} = \{Y_1, Y_2, \ldots, Y_k\}$. We now show that for each $Y_i \in \overline{\text{Dep}_{\text{FUM}}(X_+)}$, $X \leftrightarrow Y_i$ is in $(\text{FUM})^+$. 

By lemma 2.3.1.4 and FM1 $X \leftrightarrow X_F^+$. By the complementation rule $X \leftrightarrow R-X^+_F$. Thus, for the set $R-X^+_F$, which is the only element in the initial value of BASIS, $X \leftrightarrow R-X^+_F$ is in $(\text{FUM})^+$. We show, by induction, on the number of passes in the second for loop that this is true after each pass through that loop. Suppose that the claim is true after $j, j \geq 0$, passes through the loop. Let $W \leftrightarrow Z$ be the NMVD used in the next pass, and suppose that the value of BASIS is changed in the pass (otherwise there is nothing to prove). Then $W$ has to be a subset of $X$; hence $W \leftrightarrow Z$ can be augmented to $X \leftrightarrow Z$. Since, by induction hypothesis, for each set $Y_i$ in (the current value) BASIS, $X \leftrightarrow Y_i$ is in $(\text{FUM})^+$ it follows, by decomposition rule, that the claim is true for the new value of BASIS. Since $\overline{\text{Dep}_{\text{FUM}}(X_+)}$ is the final value of BASIS, therefore, for each set $Y_i \in \overline{\text{Dep}_{\text{FUM}}(X_+)}$, $X \leftrightarrow Y_i$ is in $(\text{FUM})^+$. 

Since, for each $Y_i$, $1 \leq i \leq k$, $X \leftrightarrow Y_i \in (\text{FUM})^+$, $Y_i$ can be expressed as a union of the elements of $\overline{\text{Dep}_{\text{FUM}}(X_+)}$. To conclude the proof that $\text{Dep}_{\text{FUM}}(X_+) = \overline{\text{Dep}_{\text{FUM}}(X_+)}$, we now show that each element of $\text{Dep}_{\text{FUM}}(X_+)$ is a union of the elements of $\overline{\text{Dep}_{\text{FUM}}(X_+)}$. Since both collections are partitions of $R-X^+_F$, equality follows. We construct a relation $r$ with the following properties
(a) FUM hold in $r$
(b) An NMVD $X \leftrightarrow Y$ holds in $r$ if and only if either $Y-X^+_F = \emptyset$ or $Y-X^+_F$ is a union of the elements of $\overline{\text{Dep}_{\text{FUM}}(X_+)}$. 

89
Since each of the NFD in F and each of the NMVD in M hold in r, \(F \cup M\) hold in r. Therefore, \((F \cup M)^+\) hold in r. Hence, for each \(Y \in \text{Dep}_{F \cup M}(X_+), X \rightarrow Y\) hold in r. Therefore, by (b), \(Y\) is a union of some elements of \(\text{Dep}_{F \cup M}(X_+)\).

The relation r is constructed with only data values from \((a_1, a_2, \ldots, a_k, 0, 1)\) and \(\perp\). The relation contains \(2k\) tuples \(t_1, t_2, \ldots, t_{2k}\). These tuples for \(i=1, 2, \ldots, k\) are defined as:

\[
t_{2i-1}[A] = t_{2i}[A] = a_i, A \in X,
\]

\[
t_{2i-1}[A] = 0 \text{ and } t_{2i}[A] = 1, \text{ if } A \in Y_i,
\]

and

\[
t_{2i-1}[A] = t_{2i}[A] = \perp, \text{ otherwise.}
\]

We now show that each of the NFD in F holds in r. Let \(Y \rightarrow Z \in F\). If \(Y \cap (X_+^T - X) \neq \emptyset\), then \(Y \rightarrow Z\) holds in r trivially. This is because no tuple in r is total on any attributes of \(X_+^T - X\). If \(Y \cap Y_i \neq \emptyset\) and \(Y \cap Y_j \neq \emptyset\) for \(i \neq j\), then also \(Y \rightarrow Z\) holds trivially, since there is no pair of tuples in r which are \(Y\)-total. So let \(Y \subseteq X Y_i\) for some \(Y_i\). Again if \(Y \cap Y_i \neq \emptyset\), then also \(Y \rightarrow Z\) holds, since no pair of tuples are then \(Y\)-total and equal. So we assume that \(Y \subseteq X\). Then \(Z \in X_+^T\) and \(Y \rightarrow Z\) holds in r.

Next we show that each of the NMVD in M holds in r. Let \(Y \rightarrow \rightarrow Z\) be any NMVD in M. The proof is divided in the following cases:

Case 1: \(Y \cap (X_+^T - X) \neq \emptyset\). Then \(Y \rightarrow \rightarrow Z\) holds in r trivially. This is because every tuple is not \((X_+^T X)\)-total.

Case 2: \(Y \cap Y_i \neq \emptyset\) and \(Y \cap Y_j \neq \emptyset\) for \(i \neq j\). Then also \(Y \rightarrow \rightarrow Z\) holds in r trivially since every tuple is not \(Y\)-total.

Case 3: There is one \(i\) such that \(Y \subseteq X Y Y_i\). First, we notice that
\( Y \rightarrow Z \cap X_F^+ \) holds in \( r \). Next we prove that \( Y \rightarrow Z \cap Y_j \) holds for any \( j \) by cases

(a) \( Z \cap Y_j = \emptyset \). Then \( Y \rightarrow \emptyset \) holds in \( r \) by reflexivity.

(b) \( Z \cap Y_j = Y_j \). Then, either \( Y \) is a subset of \( X \) or \( Y \cap Y_j \neq \emptyset \). For the former case, \( Y \rightarrow Y_j \) is easily verified. For the latter, the only \( Y \)-total tuple are \( t_{2i-1} \) and \( t_{2i} \). Again \( Y \rightarrow Y_j \) holds in \( r \).

(c) \( Z \cap Y \neq \emptyset \) and \( Z \cap Y_j \neq Y_j \). In this case, \( Y \cap Y_j \neq \emptyset \), for otherwise \( Y \subseteq X \) and since \( Y \subseteq X \) and the algorithm has terminated \( Z-X_F^+ \) should be expressible as the union of some elements in \( \overline{\text{Dep}_{FUM}}(X_+) \). This is a contradiction to the fact that \( Z \cap Y_j \neq \emptyset \) and \( Z \cap Y_j \neq Y_j \). Therefore, \( Y \cap Y_j \neq \emptyset \). Since \( Y \cap Y_j \neq \emptyset \), the only \( Y \)-total tuples are \( t_{2i-1} \) and \( t_{2i} \). So \( Y \rightarrow Y_j \) can be easily verified.

Therefore, \( Y \rightarrow Z \cap X_F^+ \) and \( Y \rightarrow Z \cap Y_j \) for all \( j \), \( 1 \leq j \leq k \), hold in \( r \). By the union rule \( Y \rightarrow Z \) holds in \( r \).

Now suppose \( X \rightarrow Y \) holds in \( r \) and \( Y-X_F^+ \neq \emptyset \) (otherwise the proof follows). Therefore, for each \( Y_i \in \overline{\text{Dep}_{FUM}}(X_+) \), \( X \rightarrow Y \cap Y_i \) holds in \( r \). However, \( X \) does not intersect any of the \( Y_i \) in \( \overline{\text{Dep}_{FUM}}(X_+) \), so \( X \rightarrow Y \cap Y_i \) holds in \( r \) if and only if either \( Y \cap Y_i = \emptyset \) or all of \( Y_i \). Since \( Y-X_F^+ \neq \emptyset \), it follows that \( Y-X_F^+ \) is a union of some of \( Y_i \)'s, \( Y_i \in \overline{\text{Dep}_{FUM}}(X) \). This completes the proof.

In view of the definition of dependency basis of a set of attributes \( X \) with respect to a set \( F \) of NFDs and a set \( H \) of NMVDs covering \( R-X_F^+ \), the question whether an NMVD \( X \rightarrow Y \) is in \( (FUM)^+ \) is immediate. We state this fact in the following theorem.
2.3.3.4 Theorem: Let $F$ be a set of NFDs and $M$ be a set of NMVDs. Then an NMVD $X \rightarrow Y \in (F \cup M)^+$ if and only if $Y - X_F^+$ is empty or $Y - X_F^+$ is expressible as the union of some elements in $\text{Dep}_{F \cup M}(X_+)$. 