Characterization and Armstrong relations for Degenerate Multivalued Dependencies using Formal Concept Analysis *

Jaume Baixeries[1] and José Luis Balcázar[1]

Dept. Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
c/ Jordi Girona, 1-3
08034 Barcelona
{jbaixer, balqui}@lsi.upc.es

Abstract. Functional dependencies, a notion originated in Relational Database Theory, are known to admit interesting characterizations in terms of Formal Concept Analysis. In database terms, two successive, natural extensions of the notion of functional dependency are the so-called degenerate multivalued dependencies, and multivalued dependencies proper. We propose here a new Galois connection, based on any given relation, which gives rise to a formal concept lattice corresponding precisely to the degenerate multivalued dependencies that hold in the relation given. The general form of the construction departs significantly from the most usual way of handling functional dependencies. Then, we extend our approach so as to extract Armstrong relations for the degenerate multivalued dependencies from the concept lattice obtained; the proof of the correctness of this construction is nontrivial.

1 Introduction

It is well-known [19] that, from the Concept Lattice associated to a given binary relation, one can extract a number of implications that hold in the relation, for instance via the Duquenne-Guigues basis or, alternatively, by using minimal hypergraph transversals of the predecessors of each closed set ([26], [27]). Actually, the implications obtained in that way are known to characterize a Horn axiomatization of the given relation if one exists (otherwise, they provide a well-defined Horn approximation to the data; see [3], where a complete proof may be found).

Moreover, in [19] we also find an interesting application to database theory, since the syntactical similarity between implications and functional dependencies is more than a mere syntactical similarity; and there is a precise method (that we will call here "comparison-based binarization") to change a given database relation r into a binary relation, or scaling, whose implications (its Horn axiomatization) provide exactly the functional dependencies that hold in r. Specifically,

* This work is supported in part by MCYT TIC 2002-04019-C03-01 (MOISES) and by the PASCAL-NETWORK Project

for each pair of tuples in r the values for each attribute are compared so as to yield a binary result, and therefore a binary relation (of quadratic size) is obtained.

There are other forms of dependencies in database theory, and we consider it interesting to find methods to obtain or characterize them on the basis of Formal Concept Analysis (FCA). Indeed, we have seen that this is possible for some of them, but the task turns out to be far from trivial. In [4] we have developed a careful semantic study of the relations or propositional theories where formulas of these sorts do hold: namely, multivalued dependencies, degenerate multivalued dependencies, and a family of propositional formulas introduced by Sagiv [28] that parallel them in the same sense as Horn clauses parallel functional dependencies. There we have identified precise meet operators, that is, various forms of combining objects to replace the standard intersections in the closure property, that characterize semantically these dependencies; but these do not readily give as yet a formal concept lattice. Here we consider instead an alternative approach based on defining Galois connections on classes, or partitions, of tuples and of attributes.

Along this line, in [1] we actually demonstrate how to define a Galois connection between attributes and partitions of tuples, inspired by the algorithms in [21] for computing functional dependencies, and thus propose a closure operator giving another precise characterization of functional dependencies in terms of FCA; and our recent, still unpublished work [2] proves that this approach, generalized to handle partitions of attributes instead of single attributes, actually does handle adequately multivalued dependencies.

Associated to each sort of dependency in databases is a notion of Armstrong relation: for a given set of rules, according to whatever syntax the sort of dependency considered allows for, an Armstrong relation is a relation that obeys that set of rules, and of course their logical consequences, but absolutely no other rule. They are useful in database design since they exemplify a set of dependencies, for instance those designed for a database schema prior to actually populating the database with tuples, and the analysis of such an Armstrong relation (seen as a potential database that the rules do allow, but where no other rules hold) is valuous for the prior understanding of the schema designed. Indeed, if the schema should have included a dependency that was forgotten, or expected to follow from the others, but actually does not, the Armstrong relation will point it out by including tuples that violate it, and this is usually a good method for the database engineer to realize the omission.

Our goal in this paper is twofold: first, we complete the existing characterizations by exhibiting an FCA-based characterization of degenerate multivalued dependencies, that did not follow from the previous study; and then, taking advantage of the crisper semantics of these dependencies in comparison with multivalued dependencies proper, we set out to explore the possibility of using FCA methods to construct appropriate Armstrong relations. For the case of functional dependencies there are methods actually very close to FCA [13];

in our case of DMVD's, we describe here a method to obtain the Armstrong relation from the associated Concept Lattice.

We choose a way of handling partitions of attributes and classes of tuples, and define a Galois connection between them; then we prove that the associated closure operator yields exactly the degenerate multivalued dependencies that hold in the original table, and move on to provide and fully verify a method to obtain an Armstrong relation from the family of the closed sets. Future goals would be to find a similarly-behaving method for multivalued dependencies proper, or for other stronger notions of dependency arising in the database theory area; and to investigate a potential, possibly strong connection between the notions of dependency that can be exemplified by Armstrong relations and those that can be characterized in terms of Formal Concept Analysis.

2 Basic Definitions

The following definitions and propositions have been taken from [14] and [31]. We keep mostly a database-theoretic notation since we are working for the most part with multivalued relations. We have a set of attributes $U := \{X_1, \ldots, X_n\}$ and, for each attribute $X_i \in U$, a domain of values $Dom(X_i)$. A tuple t is a mapping from U into the union of the domains, such that $t[X_i] \in Dom(X_i)$ for all i. Alternatively, tuples can be seen as well as elements of $Dom(X_1) \times \cdots \times Dom(X_n)$. We also use tuples over subsets X of U, naturally extending the notation to $t[X]$ for such projected sub-tuples. We will freely use the associativity (modulo a bijection) of the cartesian product to speak of tuples over a subset X of U as being obtained from cartesian products of subsets of X. A relation r over U is a set of tuples. We will use capital letters from the end of the alphabet for sets of attributes, and abuse language by not distinguishing single attributes from singleton sets of attributes; the context will allow always easy disambiguation. We denote by XY the union of the sets of attributes X and Y.

Given two tuples t_1, t_2, we say that they **agree** on a set of attributes X if their values in X are pointwise the same or, formally, $t_1[X] = t_2[X]$ seen as tuples.

2.1 Degenerate Multivalued Dependencies

This sort of database constraints, introduced in [15], is of intermediate strength between functional dependencies and multivalued dependencies. Whereas of less practical importance than either, it has been the right spot for generalizing the construction of Armstrong relations as we prove in the next section.

Definition 1. *A* **degenerate multivalued dependency** $X \to Y|Z$ *holds in a relation iff whenever two tuples* t_1, t_2 *agree on* X*, then, they agree on* Y *or in* Z*, where* $X \cup Y \cup Z = U$*. Formally:* $t_1[X] = t_2[X]$ *implies* $t_1[Y] = t_2[Y] \vee t_1[Z] = t_2[Z]$

The *dependency basis* for a given set of attributes $X \subseteq U$, which is a notion deeply related to multivalued dependencies in their general form, was defined in [7]. It consists in a unique partition of the set of attributes U such that, for any (nondegenerate) multivalued dependency (MVD) that has X in the left-hand side, the right-hand side is a union of one or more of the classes of this partition. It could be defined in this way because MVD's fulfill the reflexivity property ($Y \subseteq X \models X \rightarrow Y$) and their right-side parts are closed under union ($X \rightarrow Y, X \rightarrow Z \models X \rightarrow Y \cup Z$).

The dependency basis of [7] for MVD's easily allows one to summarize a set of dependencies with the same left-hand side in a single expression (as it has been proposed in [12]). Since DMVD's also have the reflexivity property and are closed under union, likewise, we summarize a set of DMVD's in a single (generalized) DMVD as follows:

Definition 2. *$X \rightarrow Y_1|\ldots|Y_n$ (where $Y_1 \cup \cdots \cup Y_n$ is a partition of U) is a* **generalized DMVD** *in r if for each DMVD $X \rightarrow Z|W$ that holds in r, Z (W) can be formed by the union of some Y_i, and $\forall Y_i : X \rightarrow Y_i|U \setminus (X \cup Y_i)$ holds in r.*

We will use here vertical bars in enumerations of classes of attributes, when they make sense syntactically as a potential generalized dependency, that is, when they constitute a partition of the attributes.

3 Modeling DMVD's with FCA

We are ready to describe the first core contribution of this paper. On the basis of an input relation r, we define a pair of functions for which we prove that they constitute a Galois connection (in a slightly generalized sense); then we analyze the closure operator obtained, and show that the implications it provides correspond to the degenerate multivalued dependencies that hold in the given r. To explain our construction, we will use throughout the following running example, in which $U = \{A, B, C, D\}$ and that has four tuples: $\{1, 2, 3, 4\}$. For sake of simplicity, we will use the tuple id's to refer the tuples:

Example 1.

id	A	B	C	D
1	a	b	c	d
2	a	b	d	c
3	a	c	c	d
4	a	c	d	b

Often Galois connections are defined between two sub-semilattices of a power set lattice; however, structurally it is not necessary to restrict ourselves to power set lattices, and many other partial orderings are as appropriate [23]. We will work with the following orderings: on the one hand, partitions of the set of attributes U, ordered according to a refinement relation; on the other hand,

sets of classes of tuples, ordered according to a covering relation similar to the refinement relation[1].

Thus, assume we are given a set of tuples r over the schema U. The central definition for our notion of closure is as follows:

Definition 3. *A partition P of the set of attributes U, $P := \{P_1|\dots|P_n\}$,* **matches** *a class of tuples $\pi \subseteq r$ iff different tuples differ on a single class. We also say that P matches a set of classes of tuples Π if P matches all classes $\pi_i \in \Pi$.*

Therefore, t_1, $t_2 \in \pi$, for a matching π, requires that there is some j such that $t_1[P_j] \neq t_2[P_j]$ but $t_1[P_k] = t_2[P_k]$ for $k \neq j$. Moreover, in principle this j depends of the pair of tuples, as the definition is stated, but in fact it does not: all the different tuples that belong to the same class agree on all the values of the different classes of attributes P_i except in one and only one sole class P_j, the same for all the pairs of tuples in π as we prove in the next proposition.

Proposition 1. *Let π be a class of tuples matched by a partition of attributes P; then all different tuples differ in the same class of attributes.*

Proof. Let us suppose that given partition of attributes $P := \{P_1|\dots|P_n\}$ there are two tuples t_1, t_2 in π such that $t_1[P_i] \neq t_2[P_i]$ and let us suppose that there is also another tuple t_3 such that $t_2[P_j] \neq t_3[P_j]$. Since each pair has to agree in the rest of the classes in which they do not differ, t_1 and t_3 will differ in P_i and in P_j, which would be a violation of the definition unless $i = j$. ∎

Intuitively, we are implicitly handling a structural binarization, or scaling, that provides us with a standard binary formal context, which in turn is able to give us the implications as we need them; the objects of this implicit scaling are the classes of tuples, of which actually only the maximal ones are relevant, whereas the implicit attributes are the partitions of the original attributes; and the binary relation is defined exactly by the *match* property. The following monotonicity property holds.

Lemma 1. *Let P, P' be two partitions of attributes, where P' is finer than (or equal to) P, that is, $\forall p' \in P' : \exists p \in P(p' \subseteq p)$. Then, each class matched by P' is also matched by P.*

Proof. Let π' be a class of tuples matched by P'. For each pair of tuples in it, there will be only one sole class of attributes from P' in which both tuples will disagree, and it will be fully contained in a class of P, which is thus the only class of P where these two tuples can disagree, which means that P will also match π'. ∎

[1] Our sets of attributes will be called classes since they belong to equivalence relations; to keep consistency, we call classes also the sets of tuples, and indeed their role is very similar to the equivalence classes of tuples in [1]. A set-theorist might object to our practice of calling sets (of classes) the third order objects and classes the second order objects, but note that in our absolutely finitary approach everything is (set-theoretic) sets and there are no proper (set-theoretic) classes.

We have a similar monotonicity property along the other dimension. To compare sets of classes of tuples, we use the following ordering:

Definition 4. *Let Π and Π' be sets of classes of tuples. $\Pi' \leq \Pi$ means: $\forall \pi' \in \Pi' : \exists \pi \in \Pi (\pi' \subseteq \pi)$.*

According to this ordering, the monotonicity property reads:

Lemma 2. *Let Π, Π' be sets of classes of tuples; if $\Pi' \leq \Pi$ and P matches Π, then P also matches Π'.*

Proof. Let t_1, t_2 be two tuples that are in the same class $\pi' \in \Pi'$. Since $\Pi' \leq \Pi$, then they will be in one class $\pi \in \Pi$, and they will disagree in only one class of attributes P_i. Therefore, P matches Π'. ∎

We denote by $\wp$ as the set of all possible partitions of attributes and present the definition of the two functions of the Galois connection.

Definition 5. *Operator ϕ: $\wp \rightarrow \mathcal{P}(\mathcal{P}(r))$. This operator receives a partition of the set of attributes U: $P := \{P_1 | \ldots | P_n\}$ and returns a set of classes of tuples $\Pi := \{\pi_1, \ldots, \pi_m\}$ matched by P, each of which is maximal under inclusion.*

Of course, $\phi(P)$ may not be a partition, in that a given tuple can belong to more that one class matched by P; this can be seen also in Example 1: $\phi(\{A|B|CD\}) = \{\{1,2\},\{3,4\},\{1,3\}\}$, and $\phi(\{A|BCD\}) = \phi(\{ABCD\}) = \{\{1,2,3,4\}\}$.

Definition 6. *Operator ψ: $\mathcal{P}(\mathcal{P}(r)) \rightarrow \wp$. This operator receives a set of classes of tuples $\Pi := \{\pi_1, \ldots, \pi_n\}$ and returns a partition of the set of attributes P that is the* **finest** *partition of attributes that matches that set of classes.*

Actually, this last definition could, in principle, be ambiguous, but fortunately there is always a unique most refined partition of attributes for a given set of classes. It is instructive to see how to prove it. We work on the basis of the following operation, which will take the role of a meet in the semilattice, and which we find most intuitive to call intersection of partitions.

Definition 7. *Let $X := \{X_1 | X_2 | \ldots | X_n\}$ and $Y := \{Y_1 | Y_2 | \ldots | Y_m\}$ be two partitions of attributes. We define the* **intersection** *of two partitions of attributes $X \cap Y$ as follows: $X \cap Y := \{Z_1 | Z_2 | \ldots | Z_l\}$ such that $\forall Z_i (\exists j, k : Z_i = X_j \cap Y_k)$.*

Example 2. Let $X := \{AB|CD|EFG\}$ and $Y := \{ABC|DEF|G\}$, then, $X \cap Y := \{AB|C|D|EF|G\}$.

It is easy to see that this operation is associative.

Lemma 3. *Let $X := \{X_1 | X_2 | \ldots | X_n\}$ and $Y := \{Y_1 | Y_2 | \ldots | Y_m\}$ two partitions of attributes, and let π be a class matched by both X and Y. Then, $X \cap Y$ also matches π.*

Proof. Let t_1, t_2 be two tuples in π. They disagree only in one class of attributes in X, let it be X_i, and in one sole class of attributes in Y, let it be Y_j. Then, the set of attributes in which both tuples disagree must be contained in both X_i and Y_j, and hence in $X_i \cap Y_j$. This proves that the intersection $X \cap Y$ also matches π. ∎

Corollary 1. *(Unicity of the most refined matcher). Let π be a class of tuples. There is a unique partition of attributes that is the most refined partition of attributes that matches π.*

Proof. The trivial partition consisting of a single class of attributes matches all classes. If two or more different and incomparable partitions of attributes match the same class, according to Lemma 3 and to the associativity of the intersection, their intersection, which is more refined, also matches π. ∎

In fact, this sort of intersection and the closure property that we have just argued correspond to the notion of intersection in the implicit binary formal context that we are using as scaling, alluded to above.

Now we can prove easily the basic property that relates both operators. To state it in the most standard terms we denote the ordering of partitions of attributes as follows:

Definition 8. *Let P, P' be partitions of a set of attributes U. We denote by $P \preceq P'$ the fact that P'* is finer than *P.*

Then we have:

Lemma 4. *Let P be a partition of attributes and Π' a set of classes of tuples. $P \preceq \psi(\Pi') \Leftrightarrow \Pi' \leq \phi(P)$.*

Proof. ($\Rightarrow$) Let us call $\psi(\Pi') = P'$. By Lemma 1, every class of tuples matched by P' is also matched by P. Then, every class of tuples matched by $\psi(\Pi')$ is included in some maximal class matched by P; that is: $\Pi' \leq \phi(P)$.

($\Leftarrow$) Let us call $\phi(P) = \Pi$. By Lemma 2, since $\Pi' \leq \Pi$ and and P matches Π (since $\phi(P) = \Pi$), then we have that P matches Π'. Since $\psi(\Pi')$ is the finest partition of attributes that matches Π', then it must be finer than (or equal to) P, that is, $P \preceq \psi(\Pi')$. ∎

It is well known that the property we have just proved characterizes Galois connections; thus,

Corollary 2. *ϕ and ψ is a* **Galois connection** *between U and $\mathcal{P}(r)$.*

We come now to the closure operator corresponding to this Galois connection, from which we will obtain the dependencies we search for, which is $\Gamma := \psi \circ \phi$. The informal definition is that given a partition of the set of attributes P, it returns the finest partition of attributes P' that can match the same classes as

P. For instance, in Example 1 we have that $\Gamma(\{B|ACD\}) = \{A|B|CD\}$ because both match the classes $\{\{1,2\},\{3,4\},\{1,3\}\}$, but $\{A|B|CD\}$ is the finest; and, also, $\Gamma(\{ABCD\}) = \{A|BCD\}$. Such a composition of both directions of a Galois connection is always a closure operator [19]. Thus,

Proposition 2. *Γ is a* **closure operator**.

We are ready for the main result in this section: the closure operator so constructed corresponds, in a precise sense, to the degenerated multivalued dependencies that hold in the original relation r. Specifically, we consider pairs of partitions of attributes having the same closure; for instance, one of them could be the closure of the other. We consider pairs of partitions that differ in only one class of attributes in the following way: we assume (to simplify notation) that they have been numbered in such a way that the first k classes in both partitions coincide, and the rest of them in X are merged into a single class in X'. We prove that, in this case, the union of the common classes can be taken as left hand side of a degenerate multivalued dependency with the remaining classes of X as right hand side, and it holds in the input relation; moreover, the converse also holds, so that each true DMVD of r gives rise in the same way to two partitions X and X' having the same closure.

Theorem 1. *Let X, X' be partitions of attributes such that the n classes of X are $X := \{X_1|\ldots|X_k|X_{k+1}|\ldots|X_n\}$, whereas the $k+1$ classes of X' are $X' := \{X_1|\ldots|X_k|X_{k+1}\ldots X_n\}$. Then, $\Gamma(X) = \Gamma(X') \Leftrightarrow X_1\ldots X_k \rightarrow X_{k+1}|\ldots|X_n$*

Proof. ($\Rightarrow$) Since $\Gamma(X) = \Gamma(X')$, then for each pair of tuples t_1, t_2 we have the following cases: (a) they both belong to the same class matched by X and X'. If this is the case, then they only disagree in one class of attributes. If this class is one $X_i : i \in \{1\ldots k\}$ then the DMVD holds because they disagree in $X_1\ldots X_k$. If this class is one $X_i : i \in \{k+1\ldots n\}$ then the DMVD holds as well. (b) they do not belong to the same class matched by X and X'. Then, they disagree in more than one class of attributes, let's say that at least they disagree in two classes. In X', at least, one of these classes must be in one $X_i : i \in \{1\ldots k\}$. Then, the same must apply in X, and, since both tuples disagree in $\{X_1|\ldots|X_k\}$ the DMVD holds as well.

($\Leftarrow$) If $X_1\ldots X_k \rightarrow X_{k+1}|\ldots|X_n$ holds, then, for each pair of tuples we can have two cases:

(a) they agree in $X_1\ldots X_k$. In this case, in order for the DMVD to hold, they must only dissagree in one partition of attributes $X_i : i \in \{k+1\ldots n\}$. In this case, they will also disagree only in one class of attributes in X and in X' and they will belong to the same class of tuples induced by X and X'.

(b) they disagree in $X_1\ldots X_k$. Then, if both tuples belong to a class matched by X, it means that they disagree in only one $X_i \in \{X_1|\ldots|X_k\}$ and that, therefore, they agree in the rest of classes. Then, they will disagree in the same sole class of attributes in X' as well, and they will belong to the same class induced by X and X'. If both tuples do not belong to a class matched by X, then it means that they disagree in, at least, two classes of attributes. Since one

of these classes will be one in $\{X_1|\ldots|X_k\}$, the other can also belong to the same subset of classes or to one $X_i : 1 \leq i \leq k$ and agree on the rest of classes. In either case, both tuples will also disagree in, at least, two classes of attributes in X' and they will not belong to the same class of tuples induced by X or X'.■

4 Calculating Armstrong relations

In this section we explain a method to construct Armstrong relations for DMVD's. As indicated above, an Armstrong relation obeys only a given set of dependencies and their logical consequences and no other dependency. In our case, there is an input relation r from which we have obtained the concept lattice via the Galois connection described in the previous section; we want to recover a relation r' only from the concept lattice, in such a way that r' satisfies exactly the same degenerate multivalued dependencies as r, which can be represented as a set of tuples or as a set of dependencies. In this section, for the sake of clarity, we will assume that r is represented as a set of tuples, but in the conclusions section we will argue that the method which is about to be presented is independent of the representation of r.

Along this section, we assume that the input relation r does not have constant attributes, that is, attributes which only take one single value along all the tuples of r. This is not a serious restriction since such attributes bear in fact no information and can be projected out and recovered later, if necessary; but, for the sake of completeness, a more detailed discussion of this case has been added to the next section of this paper.

Given a relation r, let $CL_1, CL_2, \ldots, CL_n$ be the closed partitions generated by Γ. We start with an empty relation r'. For each $CL_i : i \in \{1\ldots n\} = \{X_1|\ldots|X_m\}$ we add to r' a new pair of tuples for each $X_j : j \in \{1\ldots m\}$ as follows: all the values of X_j must be different in both tuples, and the values of the rest of classes must be the same. Also, each new value in this new pair of tuples must be different from the rest of existing values in r'. For each CL_i we will add $2m$ tuples, where m is the number of classes in that closed set.

Example 3. Let us suppose that, from a given relation r, the following closed sets are obtained: $\{A|BCD\}$ and $\{A|B|CD\}$. The Armstrong relation r' will be constructed as follows: for the closed set $\{A|BCD\}$ we will add two pairs of tuples: 1,2, that disagree in A and 3,4 that disagree in BCD. For the closed set $\{A|B|CD\}$ we will add three pairs: 5, 6, that disagree in A; 7, 8, that disagree in B; and 9, 10, that disagree in CD.

id	A	B	C	D
1	a_0	b_0	c_0	d_0
2	a_1	b_0	c_0	d_0
3	a_1	b_1	c_1	d_1
4	a_1	b_2	c_2	d_2
5	a_3	b_3	c_3	d_3
6	a_4	b_3	c_3	d_3
7	a_5	b_4	c_4	d_4
8	a_5	b_5	c_4	d_4
9	a_6	b_6	c_5	d_5
10	a_6	b_6	c_6	d_6

Theorem 2. *The method described calculates an Armstrong relation for a given set of DMVD's.*

Proof. According to Theorem 1, it is enough to prove that this relation produces the same set of closed partitions. Equivalently, that if a given partition of attributes is closed under r, it is also closed under r', and that if this partition is not closed under r it is not closed under r' either. We will prove it in two steps:

1. ($\Rightarrow$) If $X := \{X_1|\ldots|X_m\}$ is closed under r it is also closed under r'. Let's suppose that X is not closed in r'. Let Y be the closure of X in r' that matches the same classes of tuples as X. Since it is more refined, there will be two attributes that are in the same class in X, let it be X_i, and that are in different classes in Y, let it be Y_j and Y_k. By construction of r', there will be a pair of tuples in r' such that they will differ in all the attributes X_i, and they will be matched by X. These same pair of tuples will differ in classes Y_j and Y_k, and will not be matched by Y, but it contradicts our previous assumption.
2. ($\Leftarrow$) We assume that X is a partition closed in r' but not in r and derive a contradiction. Consider first the case where X contains a single class with all the attributes (the coarsest trivial partition). This X is always closed according to r', and would not be closed according to r only if some constant attribute exists in r; but we have explicitly assumed that this is not the case, so it is closed in both. For the rest of the argument, we consider some X that has at least two classes.
   Let Y be the closure of X according to r. Thus, Y is a partition that is strictly finer than X, since X is not closed and thus is different from its closure; and both X and Y match the same sets of tuples in r. Let X_i be a class of X that is not in Y, and let $Y_i \subset X_i$ a class of Y strictly included in X_i. Let W be the union of the rest of the classes of Y included in X_i, so that $X_i = Y_i \cup W$, and W is a union of classes of Y; also, $W \neq \emptyset$.
   We compare now Y_i to X_i according to r'. We have that X is closed in r', so that no further refinement is possible unless some class of tuples in r' is lost. We nevertheless refine X by splitting X_i into Y_i and W, and pick two tuples of r' that witness the fact that classes of tuples are thus lost. That

is, let t'_1 and t'_2 be tuples in r' that do match X, but would not match it anymore if X_i is split into Y_i and W.
Given that X has at least two nonempty classes, there are attributes on which t'_1 and t'_2 have the same value. These pair of tuples, that agree in several attributes, witness in r' the existence of a closure in r, by construction of r'. Let Z be this closed partition of attributes, and let $Z_i \in Z$ be the class of attributes that differ in t'_1, t'_2. On the other hand, t'_1 and t'_2 together did match X, and differ in X_i since otherwise they would still match after refining it; thus, they cannot differ anywhere else in X. This implies that all their differing attributes, namely Z_i, belong to X_i: hence $Z_i \subseteq X_i$.
Moreover, t'_1 and t'_2 differ both in some attribute of Y_i and in some attribute of W, otherwise they would still match the split. This implies that $Z_i \cap Y_i \neq \emptyset$, and that $Z_i \cap W \neq \emptyset$ as well. Note that $Z_i \subseteq X_i = Y_i \cup W$.
We return back and reason at r. Since Z is closed according to r, if we split its class Z_i into $Z_i \cap Y_i$ and $Z_i \cap W$, then some class of tuples must be lost, and, similarly to above, there must exist tuples t_1 and t_2 in r that differ in some attribute of $Z_i \cap Y_i$ and also in some attribute of $Z_i \cap W$, and coincide everywhere outside Z_i. Since $Z_i \subseteq X_i$, they coincide everywhere outside X_i as well, and thus they match X. However, they do not match Y, because they exhibit differences both in Y_i and in some attribute of W, which belongs to some other class Y_j different from Y_i. This implies that X and Y do not match the same sets of tuples of r, and therefore Y cannot be the closure of X as assumed initially. Having reached this contradiction, we have proved that X must be closed according to r if it is closed according to r'. ∎

5 Conclusions

In this paper we have used FCA to represent the set of DMVD's that hold in a relation. This can be useful because we can profit from the expressiveness of FCA (using the concept lattice as a graphical tool) and also because the different brands of dependencies so far examined can be expressed in function of a closure operator, thus complementing existing characterizations based on algebraic or logic-based approaches.

We also have presented a method for constructing an Armstrong relation following a similar approach to that used in [13] for functional dependencies, in which the notion of closed set (though not in an FCA context) was used: for each closed set, a witness was added to the Armstrong relation. It is worth to note that an Armstrong relation is usually constructed for an input relation, represented explicitly as a set of tuples or implicitly as a set of dependencies that the tuples coming in in the future will obey. Of course, when it is represented as a set of tuples, this same representation is an Armstrong relation for itself, but it does not prevent us from constructing a new one that can be much smaller in number of tuples, and more self-explanatory from the point of view of a database analyzer, for instance. In any case, the set of closures can be constructed from

the set of DMVD's according to Theorem 1 in the same way the set of closures in [13] can be extracted from a set of FD's.

5.1 Constant Attributes

Here we discuss briefly the case where the trivial coarsest partition, consisting of a single class with all the attributes, is not closed according to the input relation r. This means that it is possible to split its only class into a finer partition, and yet the set of all the tuples in r will still match, that is, all the tuples will coincide in all classes except one, always the same by Proposition 1. Everywhere outside this class, all the tuples coincide, and therefore the attributes exhibit a constant value. The closure of the trivial coarsest partition is formed thus: each such constant attribute becomes a singleton class, and the others are kept together in a single class.

These constant attributes actually bear no information at all, and can be safely projected out of r, and added to it again later if need arises. Therefore, the construction of the Armstrong relation is to be done only in the nontrivial part, so that the argument in the previous paragraphs fully applies. Alternatively, the constant attributes of r can be maintained in r', but are to be kept constant as well, instead of following the process of creation of r' that we have explained; this process would apply only to the nonconstant attributes.

5.2 Future Extensions

The next goal is to find a method, similar in spirit, to construct Armstrong relations for multivalued dependencies proper, comparing it with the one we have described here and with somewhat similar methods existing for the construction of Armstrong relations for functional dependencies [13]. On the other hand, it is known that there are notions of dependency that do not allow for Armstrong relations, and here a higher-level question arises: what is the relationship between those sorts of dependencies that have Armstrong relations and those sorts of dependencies for which a scaling exists (based maybe on partitions or sets like our own here) that provides these dependencies as the implications corresponding to the associated concept lattice. In fact, this is the major question we intend to progress on along our current research.

References

1. Baixeries J. *A Formal Concept Analysis Framework to Model Functional Dependencies.* Mathematical Methods for Learning (2004).
2. Baixeries J., Balcázar J.L. *Using Concept Lattices to Model Multivalued Dependencies.*
3. Balcázar, J.L., Baixeries J. *Discrete Deterministic Data Mining as Knowledge Compilation.* Workshop on Discrete Mathematics and Data Mining in SIAM International Conference on Data Mining (2003).

4. Balcázar, J.L., Baixeries J. *Characterization of Multivalued Dependencies and Related Expressions* Discovery Science 2004.
5. Bastide Y., Pasquier N., Taouil R., Stumme G., Lakhal L. *Mining Minimal Non-Redundant Association Rules using Closed Itemsets.* Proc. of the 1st Int'l Conf. on Computational Logic, num. 1861, Lectures Notes in Artificial Intelligence, Springer, july 2000, pages 972-986.
6. Bastide Y., Taouil R., Pasquier N., Stumme G., Lakhal L. *Mining Frequent Patterns with Counting Inference.* SIGKDD Explorations 2(2): 66-75 (2000).
7. Beeri, C. *On the Membership Problem for Functional and Multivalued Dependencies in Relational Databases.* ACM Trans. Database Syst. 5(3): 241-259 (1980)
8. Birkhoff G. *Lattice Theory, first edition.* Amer. Math. Soc. Coll. Publ. 25, Providence, R.I. 1973.
9. Davey B.A., Priestley H.A. *Introduction to Lattices and Order.* Second edition. Cambridge University Press, 1990, 2002.
10. Elmasri R., Navathe S. B. *Fundamentals of Database Systems.* 2nd Edition. Benjamin/Cummings 1994
11. Fagin R. *Functional dependencies in a relational database and propositional logic.* IBM J. Research and Development 21, 6, Nov. 1977, pp. 534-544.
12. Fagin R. *Multivalued dependencies and a new normal form for relational databases.* ACM Trans. on Database Systems 2, 3, Sept. 1977, pp. 262-278.
13. Fagin R. *Armstrong databases.* Invited paper, Proc. 7th IBM Symposium on Mathematical Foundations of Computer Science, Kanagawa, Japan, May 1982.
14. Fagin R., Beeri C., Howard J. H. *A complete axiomatization for functional and multivalued dependencies in database relations.* Jr. Proc. 1977 ACM SIGMOD Symposium, ed. D. C. P. Smith, Toronto, pp. 47-61.
15. Fagin R., Sagiv Y., Delobel D., Stott Parker D. *An equivalence between relational database dependencies and a fragment of propositional logic.* Jr. J. ACM 28, 3, July 1981, pp. 435-453. Corrigendum: J. ACM 34, 4, Oct. 1987, pp. 1016-1018.
16. Fagin R., Vardi Y. V. *The theory of data dependencies: a survey.* Mathematics of Information Processing, Proceedings of Symposia in Applied Mathematics, AMS, 1986, vol. 34, pp. 19-72.
17. Flach P., Savnik I. *Database dependency discovery: a machine learning approach.* AI Communications,volume 12 (3): 139–160, November 1999.
18. Flach P., Savnik I. *Discovery of multivalued dependencies from relations.* Intelligent Data Analysis,volume 4 (3,4): 195–211, November 2000.
19. Ganter B., Wille R. *Formal Concept Analysis. Mathematical Foundations.* Springer, 1999.
20. Godin R., Missaoui R. *An Incremental Concept Formation Approach for Learning from Databases.* Theoretical Computer Science, Special Issue on Formal Methods in Databases and Software Engineering, 133, 387-419.
21. Huhtala Y., Karkkainen J., Porkka P., Toivonen H. *TANE: An Efficient Algorithm for Discovering Functional and Approximate Dependencies.* The Computer Journal 42(2): 100 - 111, 1999.
22. Kivinen J., Mannila H. *Approximate inference of functional dependencies from relations.* Theoretical Computer Science 149(1) (1995), 129-149.
23. Liquiere M., Sallantin J. *Structural machine learning with Galois lattice and Graphs.* International Conference in Machine Learning, ICML 1998.
24. Lopes, S., Petit J-M., Lakhal L. *Efficient Discovery of Functional Dependencies and Armstrong Relations.* Proceedings of the 7th International Conference on Extending Database Technology (EDBT 2000), Konstanz, Germany.

25. Lopes, S., Petit J-M., Lakhal L. *Functional and approximate dependency mining: database and FCA points of view.* Special issue of Journal of Experimental and Theoretical Artificial Intelligence (JETAI) on Concept Lattices for KDD, 14(2-3):93-114, Taylor and Francis, 2002.
26. Pfaltz, J.L. *Transformations of Concept Graphs: An Approach to Empirical Induction.* 2nd International Workshop on Graph Transformation and Visual Modeling Techniques. GTVM 2001, Satellite Workshop of ICALP 2001, Crete, Greece. Pages 320-326. July 2001.
27. Pfaltz, J.L., Taylor, C.M. *Scientific Discovery through Iterative Transformations of Concept Lattices.* Workshop on Discrete Mathematics and Data Mining at 2nd SIAM Conference on Data Mining, Arlington. Pages 65-74. April 2002.
28. Sagiv Y. *An algorithm for inferring multivalued dependencies with an application to propositional logic.* Journal of the ACM, 27(2):250-262, April 1980.
29. Savnik I., Flach P. *Bottom-up Induction of Functional Dependencies from Relations.* Proc. of AAAI-93 Workshop: Knowledge Discovery in Databases. 1993
30. Ullman J.D. *Principles of Database and Knowledge-Base Systems.* Computer Science Press, Inc. 1988.
31. Zaniolo C., Melkanoff M. A. *On the Design of Relational Database Schemata.* TODS 6(1): 1-47 (1981).