$\mathbf{SL}_{FD}$ Logic: Elimination of data redundancy in Knowledge Representation

P. Cordero, M. Enciso, A. Mora, and I.P. de Guzmán

E.T.S.I. Informática. Universidad de Málaga. 29071. Málaga. Spain.
{pcordero,enciso}@uma.es, {amora,guzman}@ctima.uma.es

Abstract. In this paper, we propose the use of formal techniques on *Software Engineering* in two directions: 1) We present, within the general framework of lattice theory, the analysis of relational databases. To do that, we characterize the concept of f-family (Armstrong relations) by means of a new concept which we call non-deterministic ideal operator. This characterization allows us to formalize database redundancy in a more significant way than it was thought of in the literature. 2) We introduce the *Substitution Logic* SL_{FD} for functional dependencies that will allows us the design of automatic transformations of data models to remove redundancy.

Keywords: Intelligent information systems; Database and knowledge-base systems

1 Introduction

Recently, there exists a wide range of problems in Software Engineering which are being treated successfully with Artificial Intelligence (AI) techniques.Thus, [5, 6] pursue the integration between database and AI techniques, in [15, 17, 21] non classical logics are applied to *specification* and *verification* of programs, [20] shows the useful characteristics of logic for Information Systems, [11] introduces an automatic tool that translates IBM370 assembly language programs to C, etc.

Rough set theory [19] can be used to discover knowledge which is latent in database relations (e.g. data mining or knowledge discovery in database [4, 13]). The most useful result of these techniques is the possibility of "*checking dependencies and finding keys for a conventional relation with a view to using the solution in general knowledge discovery*" [3]. Moreover, in [14] the authors emphasize that the solution to this classical problem in database theory can provide important support in underpinning the reasoning and learning applications encountered in artificial intelligence. The discovery of keys can also provide insights into the structure of data which are not easy to get by alternative means.

In this point, it becomes a crucial task to have a special kind of formal language to represent data knowledge syntactically which also allows to automate the management of functional dependencies. There exists a collection of equivalent functional dependencies (FD) logics [2, 10, 16, 18, 22]. Nevertheless, none of them is appropriate to handle the most relevant problems of functional dependencies in an efficient way. The reason is that their axiomatic systems are not close to automation.

In [12, 14, 16], the authors indicate the difficulties of classical FD problems and they point out the importance of seeking efficient computational methods. In our opinion, an increasing in the efficiency of these methods might come from the elimination of redundancy in preliminary FD specification. Up to now, redundancy in FD sets was defined solely in terms of redundant FD (a FD α is redundant in a given set of FD Γ if α can be deduced from Γ). Nevertheless, a more powerful concept of FD redundancy can be defined if we consider redundancy of attributes within FDs.

In this work we present an FD logic which provides:

- New substitution operators which allows the natural design of automated deduction methods.
- New substitution rules which can be used bottom-up and top-down to get equivalents set of FD, but without redundancy.
- The FD set transformation induced by these new rules cover the definition of *second normal form*. It allows us to use substitution operators as the core of a further database normalization process.

Besides that, we introduce an algebraic framework to formalize the data redundancy problem. This formal framework allows us to uniform relational database definitions and develop the meta-theory in a very formal manner.

2 Closure Operators and Non-Deterministic Operators

We will work with posets, that is, with pairs $(A, \leq)$ where $\leq$ is an order relation.

Definition 1. *Let $(A, \leq)$ be a poset and $c : A \to A$. We say that c is a* **closure operator** *if c satisfies the following conditions:*

- *$a \leq c(a)$ and $c(c(a)) \leq c(a)$, for all $a \in A$.*
- *If $a \leq b$ then $c(a) \leq c(b)$ (c is monotone)*

*We say that $a \in A$ is c-***closed** *if $c(a) = a$.*

As examples of closure operators we have the lower closure operator[1]. Hereinafter, we will say lower closed instead of $\downarrow$-closed. Likewise, we will use the well-known concepts of $\vee$-semilattice, lattice and the concept of ideal of an $\vee$-semilattice as a sub-$\vee$-semilattice that is lower closed. Now, we introduce the notion of non-deterministic operator.

Definition 2. *Let A be a non-empty set and $n \in \mathbb{N}$ with $n \geq 1$. If $F : A^n \to 2^A$ is a total application, we say that F is a* **non-deterministic operator with arity** *n in A (henceforth,* ndo*) We denote the set* ndo*s with arity n in A by $\mathcal{N}do_n(A)$ and, if F is a* ndo*, we denote its arity by* $\mathtt{ar}(F)$.

As usual, $F(a_1, \ldots, a_{i-1}, X, a_{i+1}, \ldots, a_n) = \bigcup\limits_{x \in X} F(a_1, \ldots, a_{i-1}, x, a_{i+1}, \ldots, a_n)$.

[1] If $(U, \leq)$ is a poset, $\downarrow : 2^U \to 2^U$ is given by $X\downarrow = \bigcup\limits_{x \in X} (x] = \bigcup\limits_{x \in X} \{y \in U \mid y \leq x\}$.

As an immediate example we have that, if R is a binary relation in a non-empty set A, we can see R as an unary ndo in A where $R(a) = \{b \in A \mid (a,b) \in R\}$. We will use the following notation: $R^0(a) = \{a\}$ and $R^n(a) = R(R^{n-1}(a))$ for all $n \geq 1$. Therefore, we say that R is **reflexive** if $a \in R(a)$, for all $a \in A$, and we say that R is **transitive** if $R^2(a) \subseteq R(a)$, for all $a \in A$.[2]

Most objects used in logic or computer science are *defined inductively*. By this we mean that we frequently define a set S of objets as: "the smallest set of objects containing a given set X of atoms, and closed under a given set $\mathcal{F}$ of constructors". In this definition, the constructors are deterministic operators, that is, functions of A^n to A where A is the universal set. However, in several fields of Computer Science the ndo*s* have shown their usefulness. So, the interaction of these concepts is necessary.

Definition 3. *Let A be a poset, $X \subseteq A$ and $\mathcal{F}$ a family of* ndo*s in A. Let us consider the sets $X_0 = X$ and $X_{i+1} = X_i \cup \bigcup_{F\in\mathcal{F}} F(X_i^{\mathtt{ar}(F)})$ We define the* **nd-inductive closure of X under $\mathcal{F}$** *as $\mathcal{C}\ell_{\mathcal{F}}(X) = \bigcup_{i\in\mathbb{N}} X_i$. We say that X is* **closed for $\mathcal{F}$** *if $\mathcal{C}\ell_{\mathcal{F}}(X) = X$.*

Theorem 1. *Let $\mathcal{F}$ be a family of* ndo*s in A. $\mathcal{C}\ell_{\mathcal{F}}$ is a closure operator in $(2^A, \subseteq)$.*

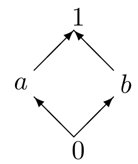
Example 1. Let $(A, \vee, \wedge)$ be a lattice. The ideal generated by X is $\mathcal{C}\ell_{\{\vee,\downarrow\}}(X)$ for all $X \subseteq A$.

3 Non-Deterministic Ideal Operators

The study of functional dependencies in databases requires a special type of ndo which we introduce in this section.

Definition 4. *Let F be an unary* ndo *in a poset $(A, \leq)$. We say that F is a* **non-deterministic ideal operator***(briefly* **nd.ideal-o***) if it is reflexive, transitive and $F(a)$ is an ideal of $(A, \leq)$, for all $a \in A$. Moreover, if $F(a)$ is a principal ideal, for all $a \in A$, then we say that F is* **principal***.*

The following example shows the independence of these properties.



Example 2. Let us consider the followings unary ndo*s* in $(A, \leq)$:

$$F(x) = \{0, x\} \qquad G(x) = \{0\} \qquad H(x) = \begin{cases} (x] & \text{if } x \neq 0 \\ A & \text{if } x = 0 \end{cases}$$

1. F is reflexive and transitive. However, F is not an nd.ideal-o because $F(1)$ is not an ideal of $(A, \leq)$.
2. G is transitive and $G(x)$ is an ideal for all $x \in A$. But, G is not reflexive.
3. H is reflexive and $H(x)$ is an ideal for all $x \in A$. However, H is not transitive because $H(H(a)) = A \not\subseteq H(a) = (a]$.

The following proposition is an immediate consequence of the definition.

[2] Or, equivalently, if $R^n(a) \subseteq R(a)$, for all $a \in A$ and all $n \in \mathbb{N} \smallsetminus \{0\}$.

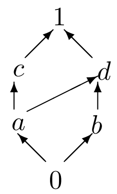
Proposition 1. *Let F be an nd.ideal-o in a poset $(A, \leq)$ and $a, b \in A$. F is a monotone operator of $(A, \leq)$ to $(2^A, \subseteq)$.*

Proposition 2. *Let $(A, \leq)$ be a lattice. The following properties hold:*

1. *Any intersection of nd.ideal-o in A is a nd.ideal-o in A.*
2. *For all unary* ndo *in A, F, there exists an unique nd.ideal-o in A that is minimal and contains $\mathcal{F}$. This nd.ideal-o is named* **nd.ideal-o generated by** *$\mathcal{F}$ and defined as $\widehat{F} = \bigcap\{F' \mid F'$ is a nd.ideal-o in A and $F \subseteq F'\}$.*[3]

Theorem 2. *Let $(A, \leq)$ be a lattice. $\widehat{\ }: \mathcal{N}do_1(A) \to \mathcal{N}do_1(A)$ is the closure operator given by $\widehat{F}(x) = \mathcal{C}\ell_{\{F,\vee,\downarrow\}}(\{x\})$.*

Example 3. Let us consider the lattice $(A, \leq)$ and the ndo given by: $F(x) = \{x\}$ if $x \in \{0, c, d, 1\}$, $F(a) = \{b, c\}$ and $F(b) = \{0\}$. Then, $\widehat{F}$ is the principal nd.ideal-o given by: $\widehat{F}(0) = \{0\}$; $\widehat{F}(b) = \{0, b\}$; $\widehat{F}(x) = A$ if $x \in \{a, c, d, 1\}$



We define the following order relation which can be read as "to have less information that".

Definition 5. *Let $(A, \leq)$ be a poset and $F, G \in \mathcal{N}do_1(A)$. We define:*

1. *$F \preccurlyeq G$ if, for all $a \in A$ and $b \in F(a)$, there exist $a' \in A$ and $b' \in G(a')$ such that $a \leq a'$ and $b \leq b'$.*
2. *$F \prec G$ if $F \preccurlyeq G$ and $F \neq G$.*

Among the generating ndo*s* of a given n.d.ideal-o we look for those that do not contain any superfluous information.

Definition 6. *Let $(A, \leq)$ be a poset and $F, G \in \mathcal{N}do_1(A)$. We say that F and G are* $\widehat{\ }$**-equivalents** *if $\widehat{F} = \widehat{G}$. We say that F is* **redundant** *if there exists $H \in \mathcal{O}nd_1(A)$ $\widehat{\ }$-equivalent to F such that $H \prec F$.*

Theorem 3. *Let $(A, \leq)$ be a poset and $F \in \mathcal{N}do_1(A)$. F is redundant if and only if any of the following conditions are fullfilled:*

1. *there exists $a \in A$ and $b \in F(a)$ such that $b \in \widehat{F_{ab}}(a)$, where F_{ab} is given by $F_{ab}(a) = F(a) \smallsetminus \{b\}$ and $F_{ab}(x) = F(x)$ otherwise.*
2. *there exists $a, b' \in A$ and $b \in F(a)$ such that $b' < b$ and $b \in \widehat{F_{abb'}}(a)$ where $F_{abb'}$ is given by $F_{abb'}(a) = (F(a) \smallsetminus \{b\}) \cup \{b'\}$ and $F_{abb'}(x) = F(x)$ otherwise.*
3. *there exists $a, a' \in A$ and $b \in F(a)$ such that $a' < a$, $b \in \widehat{F}(a')$ and $b \in \widehat{F_{aba'}}(a)$ where $F_{aba'}$ is given by $F_{aba'}(a) = F(a) \smallsetminus \{b\}$, $F_{aba'}(a') = F(a') \cup \{b\}$ and $F_{aba'}(x) = F(x)$ otherwise.*

We woulkd like to remark the fact that condition 1 is present in the database literature, but conditions 2 and 3 are stronger than it[4].

---

[3] If $F, G \in \mathcal{N}do_n(A)$ then $(F \cap G)(a) = F(a) \cap G(a)$.

[4] In fact, previous axiomatic systems can not remove redundancy from FD sets in such an easy (and automatic) way.

4 Nd.ideal-o*s* and Functional Dependencies

In this section we summarize the concepts that are basic over functional dependencies. The existence of conceptual data model with a formal basis is due, principally, to H. Codd [7]. Codd conceives stored data in **tables** and he calls *attributes* the labels of each one of the columns of the table. For each a attribute, $dom(a)$ is the domain to which the values of the column determined by such attribute belong. Thus, if $\mathcal{A}$ is the finite set of attributes, we are interested in $R \subseteq \Pi_{a\in\mathcal{A}} dom(a)$ relations. Each $t \in R$, that is, each row, is denominated *tuple of the relation.* If t is a tuple of the relation and a is an attribute, then $t(a)$ is the a-component of t.

Definition 7. *Let R be a relation over $\mathcal{A}$, $t \in R$ and $X = \{a_1, \ldots, a_n\} \subseteq \mathcal{A}$. The* **projection** *of t over X, $t_{/X}$, is the restriction of t to X. That is, $t_{/X}(a_i) = t(a_i)$, for all $a_i \in X$.*

Definition 8 (Functional Dependency). *Let R be a relation over $\mathcal{A}$. Any affirmation of the type $X{\mapsto}Y$, where $X, Y \subseteq \mathcal{A}$, is named* **functional dependency** *(henceforth FD) over R.*[5] *We say that R* **satisfies** *$X{\mapsto}Y$ if, for all $t_1, t_2 \in R$ we have that: $t_{1/X} = t_{2/X}$ implies that $t_{1/Y} = t_{2/Y}$.*
We denote by FD_R the following set $FD_R = \{X{\mapsto}Y \mid X, Y \subseteq A, R \text{ satisfies } X{\mapsto}Y\}$

In an awful amount of research on Data Bases, the study of Functional Dependencies is based on a fundamental notion: the notion of f-family (*Amstrong's Relation*) which can be characterized in the framework of the lattice theory (and without the strong restriction of working at 2^U level for a U set with finite cardinality) we present in this section.

Definition 9. *Let U be a non.empty set.*[6] *A* **f-family** *over U is a relation F in 2^U that is reflexive, transitive and satisfies the following conditions:*

1. *If $(X, Y) \in F$ and $W \subseteq Y$, then $(X, W) \in F$.*
2. *If $(X, Y), (V, W) \in F$, then $(X \cup V, Y \cup W) \in F$.*

Theorem 4. *Let A be a non-empty set and F a relation in 2^A. F is a f-family over A if and only if F is a nd.ideal-o in $(2^A, \subseteq)$.*

Proof. Let us suppose that F is a nd.ideal-o in $(2^A, \subseteq)$. If $Y \in F(X)$ and $W \subseteq Y$, since $F(X)$ is lower closed, we have that $W \in (Y] \subseteq F(X)$. Therefore, the item 1 in definition 9 is true. On the other hand, if $Y \in F(X)$ and $W \in F(V)$ then, by proposition 1, $Y \in F(X) \subseteq F(X \cup V)$ and $W \in F(V) \subseteq F(X \cup V)$. Therefore, since $F(X \cup V)$ is an $\vee$-semilattice, we have that $Y \cup W \in F(X \cup V)$ and the item 2 in definition 9 is true.

Inversely, let us suppose that F is a f-family over A and we prove that F is a nd.ideal-o in $(2^A, \subseteq)$. Since F is reflexive and transitive, we only need to probe that $F(X)$ is an ideal of $(2^{2^A}, \subseteq)$ for all $X \in 2^A$: the item 1 in definition 9 ensures that $F(X)$ is lower closed and, if we consider $V = X$, item 2 ensures that $F(X)$ is a sub-$\cup$-semilattice.

[5] This concept was introduced by Codd in 1970.
[6] In the literature, U is finite.

It is immediate to prove that, if R is a relation over A, then FD_R is a f-family (or equivalently, a nd.ideal-o in $(2^{\mathcal{A}}, \subseteq)$) The proof of the inverse result was given by Armstrong in [1]. That is, given a non-empty finite set, U, for all f-family, F, there exists a relation R (named Armstrong's relation) such that $F = FD_R$. The characterization of f-families as nd.ideal-o.s turns the proof of the well-known properties of FD_R in a trivial matter:

Proposition 3. *Let R be a relation over A. Then* [7]

1. *If $Y \subseteq X \subseteq A$ then $X \mapsto Y \in FD_R$.*
2. *If $X \mapsto Y \in FD_R$ then $X \mapsto XY \in FD_R$.*
3. *If $X \mapsto Y, Y \mapsto Z \in FD_R$ then $X \mapsto Z \in FD_R$.*
4. *If $X \mapsto Y, X \mapsto Z \in FD_R$ then $X \mapsto YZ \in FD_R$.*
5. *If $X \mapsto Y \in FD_R$ then $X \mapsto Y - X \in FD_R$.*
6. *If $X \mapsto Y \in FD_R$, $X \subseteq U \subseteq A$ and $V \subseteq XY$ then $U \mapsto V \in FD_R$.*
7. *If $X \mapsto Y, X' \mapsto Z \in FD_R$, $X' \subseteq XY$, $X \subseteq U$ and $V \subseteq ZU$ then $U \mapsto V \in FD_R$.*

Proof. Since FD_R is reflexive and lower closed, we have that $(X] \subseteq FD_R(X)$. That is, (1). Since FD_R is an $\vee$-semilattice, we have (2) and (4). Since FD_R is transitive, we have (3). Since FD_R is lower closed, we have (5).

(6): Effectively, $V \overset{1}{\in} FD_R(XY) \overset{(2)}{\subseteq} FD_R(X) \overset{(1)}{\subseteq} FD_R(U)$.

(7): Effectively, $Z \in FD_R(X') \overset{(1)}{\subseteq} FD_R(XY) \overset{(2)}{\subseteq} FD_R(X) \overset{(1)}{\subseteq} FD_R(U)$. Finally, by (2), $ZU \in FD_R(U)$) and, by (1) we have that $V \in FD_R(U)$.

5 The FDL and SL_{FD} logics

The above algebraic study and, specifically, the notion of nd.ideal-o (as an equivalent concept to the f-family concept) has guided the definition of the Functional Dependencies Logic (FDL) that we present in this section.

Definition 10. *Given the alphabet $\Omega \cup \{\mapsto\}$ where Ω is an infinite numerable set, we define the language $\mathbf{L}_{FD} = \{X \mapsto Y \mid X, Y \in 2^{\Omega}$ and $X \neq \varnothing\}$* [8].

5.1 The FDL logic

Definition 11. FDL *is the logic given by the pair $(\mathbf{L}_{FD}, \mathcal{S}_{FD})$ where $\mathcal{S}_{FD}$ has as axiom scheme $Ax_{FD}: \ \vdash_{\mathcal{S}_{FD}} X \mapsto Y$, if $Y \subseteq X$*[9]
and the following inference rules:

$(R_{\text{trans.}})$ $\quad X \mapsto Y,\ Y \mapsto Z \vdash_{\mathcal{S}_{FD}} X \mapsto Z$ **Transitivity Rule**
(R_{union}) $\quad X \mapsto Y,\ X \mapsto Z \vdash_{\mathcal{S}_{FD}} X \mapsto YZ$ **Union Rule**

In $\mathcal{S}_{FD}$ we dispose of the following derived rules (these rules appear in [18]):

$(R_{g.augm.})$ $\quad X \mapsto Y \vdash_{\mathcal{S}_{FD}} U \mapsto V$, if $X \subseteq U$ and $V \subseteq XY$
Generalized Augmentation Rule

[7] If X and Y are sets of attributes, XY denote to $X \cup Y$.
[8] In the literature, attributes must be non-empty. In FD logic, we consider the empty attribute ($\top$) to ensure that the substitution-operators introduced in section 5.2 (see definition 15) are closed.
[9] Particulary $X \mapsto \top$ is an axiom scheme.

$(R_{g.trans.})$ $X \mapsto Y,\ Z \mapsto U \vdash_{\mathcal{S}_{FD}} V \mapsto W$, if $Z \subseteq XY$, $X \subseteq V$ and $W \subseteq UV$
Generalized Transitivity Rule

The *deduction* and *equivalence* concepts are introduced as usual:

Definition 12. *Let $\Gamma, \Gamma' \subseteq \mathbf{L}_{FD}$ and $\varphi \in \mathbf{L}_{FD}$. We say that φ is deduced from Γ in $\mathcal{S}_{FD}$, denoted $\Gamma \vdash_{\mathcal{S}_{FD}} \varphi$, if there exists $\varphi_1 \dots \varphi_n \in \mathbf{L}_{FD}$ such that $\varphi_n = \varphi$ and, for all $1 \leq i \leq n$, we have that $\varphi_i \in \Gamma$, φ_i is an axiom Ax_{FD}, or it is obtained by applying the inference rules in $\mathcal{S}_{FD}$.*
*We say that Γ' is deduced of Γ, denoted $\Gamma \vdash_{\mathcal{S}_{FD}} \Gamma'$, if $\Gamma \vdash_{\mathcal{S}_{FD}} \alpha$ for all $\alpha \in \Gamma'$ and we say that Γ and Γ' are $\mathcal{S}_{FD}$-***equivalents***, denoted $\Gamma \equiv_{\mathcal{S}_{FD}} \Gamma'$, if $\Gamma \vdash_{\mathcal{S}_{FD}} \Gamma'$ and $\Gamma' \vdash_{\mathcal{S}_{FD}} \Gamma$*

Definition 13. *Let $\Gamma \subseteq \mathbf{L}_{FD}$ we define the $\mathcal{S}_{FD}$-closure of Γ, denoted $\mathcal{C}l_{FD}(\Gamma)$, as $\mathcal{C}l_{FD}(\Gamma) = \{\varphi \in \mathbf{L}_{FD} \mid \Gamma \vdash_{\mathcal{S}_{FD}} \varphi\}$*

Now is evident the following result.

Lemma 1. *Let Γ and $\Gamma' \subseteq \mathbf{L}_{FD}$. Then, Γ and Γ' are $\mathcal{S}_{FD}$-***equivalentes*** if and only if $\mathcal{C}l_{FD}(\Gamma) = \mathcal{C}l_{FD}(\Gamma')$.*

5.2 The logic SL_{FD}

Although the system $\mathcal{S}_{FD}$ is optimal for meta-theoretical study, in this section, we introduce a new axiomatic system (SL_{FD}) for $\mathbf{L}_{FD}$ more adequate for the applications. First, we define two substitution operators and we illustrate their behaviour for removing redundancy.
Note that the traditional axiomatic system treats the redundancy type described in the item 1. of the theorem 3. We treat in a way efficient, the redundancy elimination described in item 2 and 3 of theorem 3.

Definition 14. *Let $\Gamma \subseteq \mathbf{L}_{FD}$, and $\varphi = X \mapsto Y \in \Gamma$. We say that φ is* **superfluous in** *Γ if $\Gamma \backslash \{\varphi\} \vdash_{FD} \varphi$. We say that φ is* **l-redundant in** *Γ if exists $\varnothing \neq Z \subseteq X$ such that $(\Gamma \backslash \varphi) \cup \{(X - Z) \mapsto Y\} \vdash_{\mathcal{S}_{FD}} \varphi$. We say that φ is* **r-redundant in** *Γ if exists $\varnothing \neq U \subseteq Y$ such that $(\Gamma \backslash \varphi) \cup \{X \mapsto (Y - U)\} \vdash_{\mathcal{S}_{FD}} \varphi$. We say that Γ* **have redundancy** *if it have an element φ that it is superfluous or it is l-redundant or it is r-redundant in Γ.*

The operators that we will introduce are transformations of $\mathcal{S}_{FD}$-*equivalence*. This way, the application of this operators does not imply the incorporation of *wff*, but the substitution of *wffs* for simpler ones, with an efficiency improvement [10].

Theorem 5. *Given $X, Y, Z \in 2^{\Omega}$,*

$$\{X \mapsto Y\} \equiv_{\mathcal{S}_{FD}} \{X \mapsto (Y - X)\} \text{ and } \{X \mapsto Y, X \mapsto Z\} \equiv_{\mathcal{S}_{FD}} \{X \mapsto YZ\}$$

The following theorem allow us to introduce the substitution operators.

Theorem 6. *Let $X \mapsto Y, U \mapsto V \in \mathbf{L}_{FD}$ with $X \cap Y = \varnothing$.*

[10] It is easily proven that the reduction rule and union rule are $\mathcal{S}_{FD}$-*equivalence* transformations.

(a) *If $X \subseteq U$ then $\{X \mapsto Y, U \mapsto V\} \equiv_{\mathcal{S}_{FD}} \{X \mapsto Y, (U-Y) \mapsto (V-Y)\}$. Therefore, if $U \cap Y \neq \varnothing$ or $V \cap Y \neq \varnothing$ then $U \mapsto V$ is l-redundant or r-redundant in $\{X \mapsto Y, U \mapsto V\}$, respectively.*

(b) *If $X \not\subseteq U$ and $X \subseteq UV$ then $\{X \mapsto Y, U \mapsto V\} \equiv_{\mathcal{S}_{FD}} \{X \mapsto Y, U \mapsto (V-Y)\}$. Therefore, if $V \cap Y \neq \varnothing$ then $U \mapsto V$ is r-redundant in $\{X \mapsto Y, U \mapsto V\}$.*

Proof. **(a)**

$\Rightarrow$) : [11]		$\Leftarrow$) :	
1. $X \mapsto Y$	Hypothesis	1. $U \mapsto X$	Ax_{FD}
2. $(U\text{-}Y) \mapsto Y$	1, $R_{g.augm.}$	2. $X \mapsto Y$	Hypothesis
3. $(U\text{-}Y) \mapsto (U\text{-}Y)$	Ax_{FD}	3. $U \mapsto Y$	$1, 2, R_{trans.}$
4. $(U\text{-}Y) \mapsto UY$	$2, 3,\ R_{union}$	4. $(U\text{-}Y) \mapsto (V\text{-}Y)$	Hypothesis
5. $(U\text{-}Y) \mapsto U$	4, $R_{g.augm.}$	5. $U \mapsto VY$	$3, 4, R_{union}$
6. $U \mapsto V$	Hypothesis	6. $U \mapsto V$	$2, 5, R_{g.augm.}$
7. $(U\text{-}Y) \mapsto V$	$5, 6,\ R_{trans.}$		
8. $(U\text{-}Y) \mapsto (V\text{-}Y)$	7, $R_{g.augm.}$		

(b)

$\Rightarrow$) :		$\Leftarrow$) :	
1. $U \mapsto V$	Hypothesis	1. $U \mapsto X$	Ax_{FD}
2. $U \mapsto (V-Y)$	1, $R_{g.augm.}$	2. $X \mapsto Y$	Hypothesis
		3. $U \mapsto Y$	$1, 2, R_{trans.}$
		4. $U \mapsto (V\text{-}Y)$	Hypothesis
		5. $U \mapsto VY$	$3, 4, R_{union}$
		6. $U \mapsto V$	$2, 5, R_{g.augm.}$

The above theorems allow us to define two substitution operators as follows:

Definition 15. *Let $X \mapsto Y \in \mathbf{L}_{FD}$, we define $\Phi_{X \mapsto Y}, \Phi^r_{X \mapsto Y} : \mathbf{L}_{FD} \longrightarrow \mathbf{L}_{FD}$, denominated respectively $(X \mapsto Y)$-**substitution operator**, and $(X \mapsto Y)$-**right- substitution operator** (or simply $(X \mapsto Y)$-r-substitution operator):*

$$\Phi_{X \mapsto Y}(U \mapsto V) = \begin{cases} (U-Y) \mapsto (V-Y) & \text{if } X \subseteq U \text{ and } X \cap Y = \varnothing \text{ [12]} \\ U \mapsto V & \text{otherwise} \end{cases}$$

$$\Phi^r_{X \mapsto Y}(U \mapsto V) = \begin{cases} U \mapsto (V-Y) & \text{if } X \not\subseteq U, X \cap Y = \varnothing \text{ and } X \subseteq UV \\ U \mapsto V & \text{otherwise} \end{cases}$$

Now, we can define a new axiomatic system, $\mathcal{S}_{FDS}$, for $\mathbf{L}_{FD}$ with a substitution rule as primitive rule.

Definition 16. *The system $\mathcal{S}_{FDS}$ on $\mathbf{L}_{FD}$ has one axiom scheme:*

Ax_{FDS} : $\vdash X \mapsto Y$, *where $Y \subseteq X$. Particulary, $X \mapsto \top$ is an axiom scheme. The inferences are the following:*

$(R_{\text{frag.}})$ $\quad X \mapsto Y \vdash_{\mathcal{S}_{FDS}} X \mapsto Y'$, *if* $Y' \subseteq Y$ **Fragmentation rule**

$(R_{\text{comp.}})$ $\quad X \mapsto Y,\ U \mapsto V \vdash_{\mathcal{S}_{FDS}} XU \mapsto YV$ **Composition rule**

$(R_{\text{subst.}})$ $\quad X \mapsto Y,\ U \mapsto V \vdash_{\mathcal{S}_{FDS}} (U\text{-}Y) \mapsto (V\text{-}Y)$, *if* $X \subseteq U$, $X \cap Y = \varnothing$ **Substitution rule**

---

[11] In 2 we use $X \subseteq U - Y$ and in 4 we use $Y(U-Y) = UY$.

[12] Notice that $V - Y$ may be $\top$. In this case we will remove the *wff* using axiom Ax_{FD}.

Theorem 7. *The $\mathcal{S}_{FD}$ and $\mathcal{S}_{FDS}$ systems on $\mathbf{L}_{FD}$ are equivalent.*

Proof. Let R_{union} be a particular case of $R_{comp.}$, then all we have to do is to prove that ($R_{trans.}$) is a derived rule of $\mathcal{S}_{FDS}$:

1. $X\mapsto Y$	Hypothesis	6. $XY\mapsto(Z-Y)$	4, 5, $R_{subst.}$
2. $Y\mapsto Z$	Hypothesis	7. $(Y-X)\mapsto(Y-X)$	Ax_{FDS}
3. $X\mapsto(Y-X)$	1, $R_{frag.}$	8. $X\mapsto(Z-Y)$	3, 6, $R_{subst.}$
4. $Y\mapsto(Z-Y)$	2, $R_{frag.}$	9. $X\mapsto ZY$	1, 7, $R_{comp.}$
5. $X\mapsto\top$	Ax_{FDS}	10. $X\mapsto Z$	9, $R_{frag.}$

The example 4 shows the advantages of the Φ and Φ^r operators, and the example 5 show how is possible to automate the redundance remove process.

Example 4. Let $\Gamma=\{ab\mapsto c, c\mapsto a, bc\mapsto d, acd\mapsto b, d\mapsto eg, be\mapsto c, cg\mapsto bd, ce\mapsto ag\}$. We apply the Φ, and Φ^r for obtaining a non redundant *wffs* set and equivalent to Γ. In the following table, we show by rows how we obtain successively equivalent *wff* sets, but with less redundancy. We emphasize with $\sqcup$ both *wffs* that allow to apply the operator. We cross out with $\diagup$ the removed *wff* after the application of the operator. We remark in each row the operator or the rule applied.

$\Phi_{c\mapsto a}(acd\mapsto b)$	$\{ab\mapsto c, \underbracket{c\mapsto a}, bc\mapsto d, \underbracket{\cancel{acd\mapsto b}}, d\mapsto eg, be\mapsto c, cg\mapsto bd, ce\mapsto ag\}$
$\Phi_{c\mapsto a}(ce\mapsto ag)$	$\{ab\mapsto c, \underbracket{c\mapsto a}, bc\mapsto d, cd\mapsto b, d\mapsto eg, be\mapsto c, cg\mapsto bd, \underbracket{\cancel{ce\mapsto ag}}\}$
$\Phi^r_{bc\mapsto d}(cg\mapsto bd)$	$\{ab\mapsto c, c\mapsto a, \underbracket{bc\mapsto d}, cd\mapsto b, d\mapsto eg, be\mapsto c, \underbracket{\cancel{cg\mapsto bd}}, ce\mapsto g\}$
	$\Gamma'=\{ab\mapsto c, c\mapsto a, bc\mapsto d, cd\mapsto b, d\mapsto eg, be\mapsto c, cg\mapsto b, ce\mapsto g\}$

Example 5. Let Γ the FD set showed in the first row of the table.

$\Phi_{b\mapsto c}(bc\mapsto de)+R_{union}$	$\Gamma=\{a\mapsto b, \underbrace{b\mapsto c}, ae\mapsto cfh, \underbrace{\cancel{bc\mapsto de}}, bd\mapsto ce, afh\mapsto ce, bcd\mapsto aef\}$
$\Phi_{b\mapsto cde}(bd\mapsto ce)$[13]	$\Gamma=\{a\mapsto b, \underbrace{b\mapsto cde}, ae\mapsto cfh, \underbrace{\cancel{bd\mapsto ce}}, afh\mapsto ce, bcd\mapsto aef\}$
$\Phi_{b\mapsto cde}(bcd\mapsto aef)+R_{union}$	$\Gamma=\{a\mapsto b, \underbrace{b\mapsto cde}, ae\mapsto cfh, afh\mapsto ce, \underbrace{\cancel{bcd\mapsto aef}}\}$
$\Phi^r_{ae\mapsto cfh}(b\mapsto acdef)$	$\Gamma=\{a\mapsto b, \underbrace{\cancel{b\mapsto acdef}}, \underbrace{ae\mapsto cfh}, afh\mapsto ce\}$
$\Phi^r_{ae\mapsto cfh}(afh\mapsto ce)$	$\Gamma=\{a\mapsto b, b\mapsto ade, \underbrace{ae\mapsto cfh}, \underbrace{\cancel{afh\mapsto ce}}\}$
$R_{g.trans.}$	$\Gamma=\{\underbrace{a\mapsto b}, \underbrace{b\mapsto ade}, ae\mapsto cfh, \underbrace{\cancel{afh\mapsto e}}\}$
	$\Gamma'=\{a\mapsto b, b\mapsto ade, ae\mapsto cfh\}$

Due to space limitations, we can not go further into comparison with other axiomatic systems, nevertheless we would like to remark that $\mathcal{S}_{FDS}$ allow us to design (in a more direct way) an automated and efficient method which remove redundancy efficiently. In this case, the example 4 taken from [22] requires the application of seven $\mathcal{S}_{FD}$ rules in a non deterministic way.

[13] Notice that the Φ operator renders $b\mapsto\top$ and we remove it by using axiom Ax_{FDS}.

References

1. William W. Armstrong. Dependency structures of data base relationships. *Proc. IFIP Congress. North Holland, Amsterdam*, pages 580–583, 1974.
2. Paolo Atzeni and Valeria De Antonellis. Relational Database Theory. *The Benjamin/Cummings Publishing Company Inc.*, 1993.
3. D.A. Bell. From data properties to evidence. *IEEE Transactions on Knowledge and Data Engireering*, 5(6):965–968, 1993.
4. D.A. Bell and J.W. Guan. Computational methods for rought classifications and discovery. *J. American Society for Information Sciences*, To appear. Special issue on Data Mining.
5. Elisa Bertino, Barbara Catania, and Gian Piero Zarri. Intelligent Database Systems. *ACM Press. Addison-Wesley*, ISBN 0-201-87736-8, 2001.
6. L. E. Bertossi and J. C. Ferretti. SCDBR: A reasoner for specification in the situation calculus of database updates. In *1st International Conference on Temporal Logic*, Berlín, 1994.
7. Edgar F. Codd. The relational model for database management: Version 2. reading, mass. *Addison Wesley*, 1990.
8. Manuel Enciso and Angel Mora. FD3: A functional dependencies data dictionary. *Proceedings of the Fourth Conference on Enterprise Information Systems (ICEIS). Ciudad Real, Spain*, 2:807–811, 2002 Apr.
9. Manuel Enciso, Carlos Rossi, Juan Manuel Frias, and Angel Mora. Logic modeling of cooperative database systems. *Information Resources Management Association (IRMA)*, IDEA Group Publishing, 1999.
10. Ronald Fagin. Functional dependencies in a relational database and propositional logic. *IBM. Journal of research and development.*, 21 (6):534–544, 1977.
11. Yishai A. Feldman and Doron A. Friedman. Portability by automatic translation: A large-scale case study. *Artificial Intelligence*, 107(1):1–28, 1999.
12. Rusell Greiner. Finding optimal derivation strategies in redundant knowledge bases. *Artificial Intelligence*, 50(1):95–115, 1991.
13. J. W. Guan and D. A. Bell. Rough computational methods for information systems. *Artificial Intelligence*, 105:77–103, 1998.
14. J.W. Guan and D.A. Bell. Rough computational methods for information systems. *Artificial Intelligence*, 105:77–103, 1998.
15. Erika Hajnicz. Time structures. formal description and algorithmic representation. In *Lecture Notes in Artificial Intelligence. Num. 1047.* Springer-Verlag, 1996.
16. Toshihide Ibaraki, Alexander. Kogan, and Kazuhisa Makino. Functional dependencies in horn theories. *Artificial Intelligence*, 108:1–30, 1999.
17. Zohar Manna and A. Pnueli. *Temporal verification of reactive systems: Safety.* Springer-Verlag, 1995.
18. Jan Paredaens, Paul De Bra, Marc Gyssens, and Dirk Van Van Gucht. The structure of the relational database model. *EATCS Monographs on Theoretical Computer Science*, 1989.
19. Z. Pawlak. *Rough Set: teoretical aspects of reasoning about data.* Kluwer, 1991. Dordercht, Netherlands.
20. David Robertson and Jaum Agustí. Lightweight uses of logic in conceptual modelling. *Software Blueprints. ACM Press. Ed. Addison Wesley*, 1999.
21. Alexander Tuzhilin. Templar: a knowledge-based language for software specifications using temporal logic. *ACM Transactions on Information Systems*, 13:269–304, July 1995.
22. Jeffrey D. Ullman. Database and knowledge-base systems. *Computer Science Press*, 1988.