$\mathbf{SL}_{FD}$ Logic: Elimination of Data Redundancy in Knowledge Representation

Pablo Cordero, Manolo Enciso, Angel Mora, and Inmaculada 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.

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 [14,16,20] non classical logics are applied to *specification* and *verification* of programs, [19] shows the useful characteristics of logic for Information Systems, [10] introduces an automatic tool that translates IBM370 assembly language programs to C.

Rough set theory [18] can be used to discover knowledge which is latent in database relations (e.g. data mining or knowledge discovery in database [4,12]). 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 [13] 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 [2,9,15,17,21]. 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.

F.J. Garijo, J.C. Riquelme, and M. Toro (Eds.): IBERAMIA 2002, LNAI 2527, pp. 141–150, 2002.

In [11,13,15], 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, $F(a_1, \ldots, a_{i-1}, X, a_{i+1}, \ldots, a_n) = \bigcup_{x \in X} F(a_1, \ldots, a_{i-1}, x, a_{i+1}, \ldots, a_n)$.

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