Towards declarative comparabilities: application to functional dependencies

Lhouari Nourine[1], Jean-Marc Petit[2], and Simon Vilmin[3]

[1]Université Clermont-Auvergne, CNRS, Mines de Saint-Étienne, Clermont-Auvergne-INP, LIMOS, 63000 Clermont-Ferrand, France.
[2]INSA Lyon, CNRS, UCBL, Centrale Lyon, Univ Lyon 2, LIRIS, UMR5205, F-69621 Villeurbanne, France
[3]Aix-Marseille Université, CNRS, LIS, Marseille, France.

April 30, 2024

Abstract

In real life, data are often of poor quality as a result, for instance, of uncertainty, mismeasurements, missing values or bad inputs. This issue hampers an implicit yet crucial operation of every database management system: equality testing. Indeed, equality is, in the end, a context-dependent operation with a plethora of interpretations. In practice, the treatment of different types of equality is left to programmers, who have to struggle with those interpretations in their code. We propose a new lattice-based declarative framework to address this problem. It allows specification of numerous semantics for equality at a high level of abstraction. To go beyond tuple equality, we study functional dependencies (FDs) in the light of our framework. First, we define abstract FDs, generalizing classical FDs. These lead to the consideration of particular interpretations of equality: realities. Building upon realities and possible/certain answers, we introduce possible/certain FDs and give some related complexity results.

1 Introduction

Given two values u and v, how can we decide whether or not u and v are equal? The question is simple to ask, but much harder to answer as there exist countless possible interpretations of equality. Moreover, it is an implicit but mainstream operation in every database management system (DBMS) as well as in most database applications among which query answering [AHV95, AL18, GPW14], data integration [Len02], inconsistent databases [Ber11], probabilistic data [SORK11], data profiling [AGNP18], and database design [LP19].

At the same time, the quality of real life data is (very) often quite poor [BCFM09]. This is due to issues such as uncertainty, measurement errors, bad input or missing values to mention but a few. In those contexts, equality testing becomes even more crucial for dealing with data quality issues.

Finally, domain experts are usually the only ones who can specify what *"equality"* truly means on their data. As a consequence, the exact meaning of equality turns out to be plural and context-dependent, as many interpretations are indeed possible. In practice, implementation is left to DBMS programmers, who have to struggle with all those different interpretations of equality in their (SQL) code.

To cope with this issue, we introduce a new lattice-based declarative framework for relational databases. It allows specification of different meanings of equality at a high level of abstraction. As a consequence, we consider equality testing to be a first-class citizen. The notion of comparabilities turns out to play a major role in our framework, defined at attribute level by *comparability functions* and *abstract lattices*. Different *interpretations* are then possible as per the usual logic, making it possible to decide whether two values (or two tuples) are indeed equal.

One of the motivation for our work draws from a collaboration with an industrial partner specialized in cold chain, refrigereation and conditioning (CEMAFROID). The aim was to predict the ageing of refrigerating transport vehicles from their data [LGCP+21]. Within this collaboration, a finer consideration of equality was a key notion to select relevant data in SQL in order to get better results. Continuing this collaboration, we used the framework we introduce in this paper (with comparability functions, abstract values and lattices) to model 20 important attributes at a high level of abstraction. A (preliminary) prototype has been implemented on top of PostgreSQL with two main components: first, a concrete language on top of the DDL of PosgreSQL and second, a SQL fragment to express simple selection queries with equality.

In this paper, we focus instead on its impact on functional dependencies (FDs). We define *abstract FDs* on top of the lattice-based framework. Abstract FDs lead to the consideration of particular interpretations, called realities. Inspired by possible/certain query answering in databases [Lib16], as well as weak/strong or possible/certain FDs [AAS22, LL98], we use realities to define similar notions on functional dependencies, namely possible and certain FDs. We give some complexity results about possible/certain FDs.

To deal with missing or uncertain informations, many works have introduced new types of dependencies (see e.g. [BCKN18, BKL13, CDP15, LP19, Ng01]). In these approaches, the comparison of two values is a one-step process which returns a truth value of the underlying logic (usually the classic binary logic or the real interval $[0, 1]$). Our framework however replaces the operation of testing the equality of two values by a two-step process: comparison and interpretation. Moreover, it does not modify the input data, contrary to common approaches for dealing with missing information [AAS22, KLLZ16, KL18, LL98, Lie82]. Consequently, our framework supplies experts with the possibility to declare meaningful comparability functions and different semantics for equality separately, without having to modify their data.

Another well-studied way of dealing with uncertainty is to extend the classical relational approach to a new system based on more elaborated logics such as fuzzy relational systems (see e.g. [BV18, BPU99, JCE17, PT84, YOJ99]). Nevertheless, unlike fuzzy systems, our framework relies on the classical relational model and its underlying binary logic. Thus, the dependencies we consider are different in nature from the generalization of the classical FDs to the fuzzy set up. Moreover, since our framework is based on the usual relational model, it can come on top of any DBMS and does not require the implementation of a whole new system.

Contributions. We summarize the main contributions of our paper:

- We introduce a declarative framework at database scheme level to consider different interpretations of equalities as first-class citizens It is based on three concepts applied attribute-wise: comparability functions, abstract lattices, and interpretations.
- We apply our framework to functional dependencies:
  - We show that any relation can be associated with an *abstract lattice*, thus giving a semantic for *abstract functional dependencies*. We prove that, under certain circumstances, an interpretation preserves the validity of classical FDs for any abstract lattice. These interpretations are called realities and strong realities.
  - We investigate the plausibility of a given functional dependency: whether it is *possible*, i.e., there exists at least one reality for which the FD holds, or whether it is *certain*, i.e., holds for every reality. We also discuss this problem for the case of strong realities and show that deciding strong possibility is NP-complete.

Paper organization. In Section 2, we present our framework informally on a running example. Section 3 recalls standard notions from database and lattice theory. We formally introduce our lattice-based framework in Section 4. In Section 5, we study functional dependencies in our framework. In Section 6, we discuss related works on handling incomplete and unclean information in databases. At last, Section 7 concludes with open questions for further research.

2 Framework in a nutshell

In this section, we describe our framework using the following running example.

Example 1. (Running example) A doctor is interested in the triglyceride level (mmol/L), the gender (M/F), and the waist size (cm) of her patients. We denote these attributes by A, B, C respectively. A sample is represented in Table 1 as the relation Patients. It contains a missing value, outliers, and some very similar values.

Patients	T. level (A)	Gender (B)	W. size (C)
t_1	1.4	F	73
t_2	1.5	F	`null`
t_3	3.2	M	72
t_4	3.5	F	73
t_5	40	F	100

Table 1: Running example on medical data

Each attribute is first equipped with a *comparability function* that maps every pair of values in the attribute domain to an *abstract value* in a fine-grained similarity scale, ordered in an *abstract lattice*. A comparability function must be associative. Both the comparability function and the abstract lattice can be given at database design time.

Example 2. (Continued) Consider the waist size. Based on her background knowledge, the doctor supplies conditions to compare different waist sizes. First, she provides the main comparability values (abstract values), namely *close*, *unknown*, and *distant*. Then,

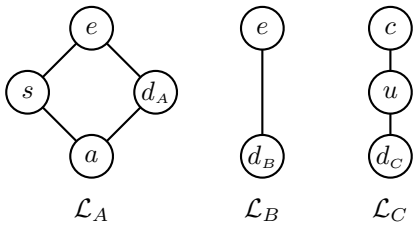


Figure 1: Abstract lattices

she uses these abstract values to define an appropriate way to compare two waist sizes (x and y are waist sizes):

- *close* if $x = y \neq \texttt{null}$ or $x, y \in [70, 80]$,
- *unknown* if $u = \texttt{null}$ or $v = \texttt{null}$,
- *distant* otherwise.

Eventually, she has to order the abstract values according to the principle *"the higher the closer"*. In the case of waist size, she ends up with the abstract lattice depicted in Figure 1 by the lattice $\mathcal{L}_C$. For example, 72 and 73 are *equal* while 73 and 100 are *distant*. More generally, we assume that the following comparability functions are defined, along with their associated abstract lattices (see Figure 1).

$$f_A(x, y) = \begin{cases} \textit{equal} & \text{if } x = y \text{ or } x, y \in [0, 2[\\ \textit{second class} & \text{if } x, y \in [2, 5[, x \neq y \\ \textit{distributed} & \text{if } (x, y) \text{ or } (y, x) \in [0, 2[\times[2, 5[\\ \textit{abnormal} & \text{otherwise.} \end{cases}$$

$$f_B(x, y) = \begin{cases} \textit{equal} & \text{if } x = y \\ \textit{different} & \text{if } x \neq y. \end{cases}$$

$$f_C(x, y) = \begin{cases} \textit{close} & \text{if } x = y \neq \texttt{null} \text{ or } x, y \in [70, 80] \\ \textit{unknown} & \text{if } u = \texttt{null} \text{ or } v = \texttt{null} \\ \textit{distant} & \text{otherwise.} \end{cases}$$

We use abbreviations for each value: e stands for *equal*, s for *second class*, a for *abnormal*, u for *unknown*, c for *close*, and d_A for *distributed*, d_B for *different* and d_C for *distant*.

Remark that all three comparability functions extend classical equality since every value, except `null`, compared with itself gives the highest possible abstract value. Then, a comparison of two tuples with this framework yields an *abstract tuple*. This is a precise account of their agreement on each attribute.

Example 3. (Continued) Consider the tuples t_1, t_2 of our running example. Let f_R be the comparability function consisting of the attribute-wise application of f_A, f_B and f_C. The comparison of t_1 and t_2 is $f_R(t_1, t_2) = \langle f_A(1.4, 1.5), f_B(F, F),\ f_C(73, \texttt{null})\rangle = \langle e, e, u\rangle$.

Abstract tuples may not be intuitive enough to be used by practitioners. Thus, the next step is to use *interpretations* to decide whether or not an abstract value is considered to be equality. At attribute level, an interpretation maps an abstract value to 0 or 1. As usual, 0 means difference, and 1 means equality. We further require that an interpretation is increasing: the higher the abstract value in the lattice, the more it gets close to equality. Thus, an interpretation is a semantic for equality on a attribute. At the scheme level, an interpretation states whether or not two tuples are considered equal.

Example 4. (Continued) The doctor gives several hypotheses on the meaning of equality with respect to her data. These lead to three possible interpretations g_1, g_2, g_3 which are represented in Figure 2. Above the dotted lines, the abstract values are interpreted as 1, and 0 otherwise. With g_1 and g_2, t_1 and t_2 are not equal, while they are with g_3 (we have $g_3(\langle e, e, u\rangle) = \langle 1, 1, 1\rangle$).

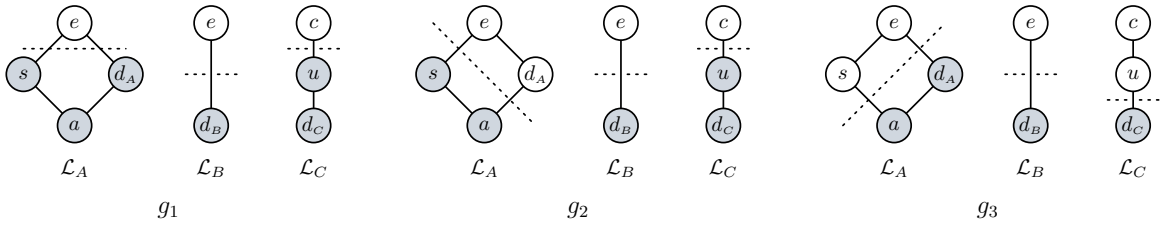


Figure 2: Three possible interpretations of similarity values: 1 above the dotted line, 0 below.

Quite clearly, this framework provides domain experts with an opportunity to specify at a high level of abstraction different types of equality on top of database schemes, independently from the underlying applications and without updating the data. We summarize our framework with the pipeline presented in Figure 3, applied to the tuples t_1, t_4 of Patients.

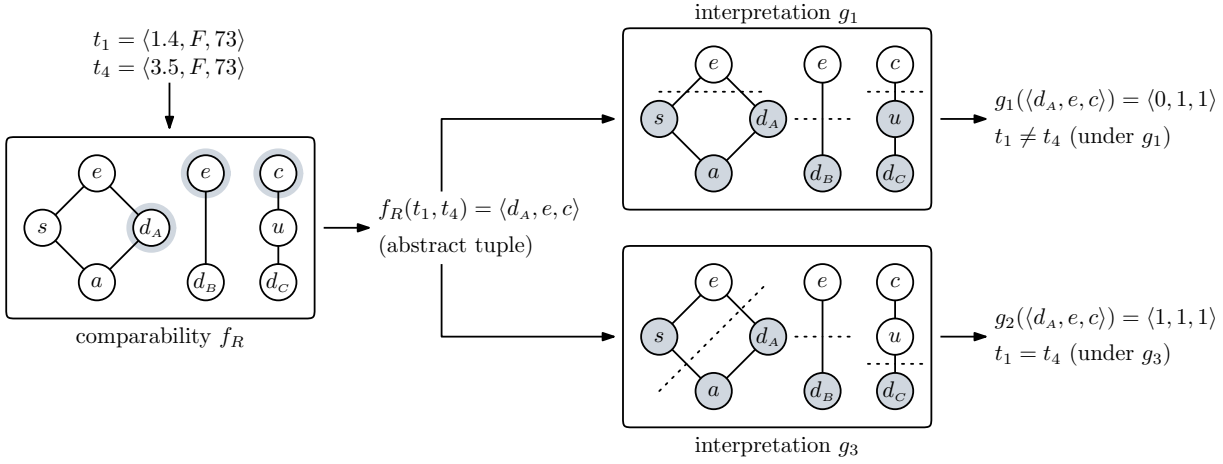


Figure 3: Pipeline of the framework on t_1, t_4.

Application to functional dependencies. Usually, a functional dependency $X \to Y$ holds in a given relation if two tuples that are equal on X are also equal on Y. This definition smoothly adapts to our framework using interpretations. More precisely, $X \to Y$

holds with respect to a given interpretation g if two tuples that are considered equal on X with respect to g, are also considered equal on Y with respect to g.

From a more theoretical point of view, we show that a relation satisfies a more general type of dependencies based on abstract tuples, independently from any interpretation. These dependencies are called *abstract functional dependencies*. They are lattice implications [Day92] of the form $x \to y$, where x and y are abstract tuples drawn from the product of abstract lattices. In the manner of classical functional dependencies and agree sets [BDFS84], the abstract functional dependencies satisfied by a relation capture the abstract knowledge of this relation, being a collection of abstract tuples ordered in a new abstract lattice. In this context, an interpretation turns the abstract lattice into a set system over the relation scheme. This separates the interpretations: some of them preserve the semantic of classical FDs, i.e. the abstract lattice is interpreted as a closure system [DP02], and some do not. We call *realities* these semantic preserving interpretations. It turns out that realities are precisely the interpretations that are lattice meet-homomorphisms [DP02]. As a side effect, the equality in realities can be represented by a unique minimum element in each attribute's abstract lattice. We also study *strong realities*, the interpretations that are lattice homomorphisms.

Building upon realities, we introduce the notion of plausibility (certain/possible) of a given FD, inspired from query answering [AL18, Ber11, Lib16] and weak/strong or possible/certain FDs [LL98, AAS22]. More precisely, the existence of numerous realities leads to cases of equality semantic where the FD holds, and some others where it does not. In this case, the FD is *possible*. In practice, this may be the case if, for instance, the relation suffers from unclean data: a FD should theoretically hold with respect to background knowledge, but due to mismeasurements or imprecisions, it fails to do so. Here, weakening/strengthening the meaning of equality according to expert knowledge could easily fix this problem, by simply applying the *"right"* reality with respect to the background knowledge. After possibility, it is also natural to wonder for a given FD if there is a chance that it holds in every reality, in which case we say that it is *certain*. It is worth noticing that the data are left unchanged in our framework. Hence, the terms *"possible"* and *"certain"* differ from the more usual point of view where a relation with `null` values is subject to numerous completions.

Example 5. (Continued) The doctor knows from her background knowledge that the triglyceride level is determined by waist size and gender. This can be modeled by the functional dependency $BC \to A$. Among the realities given in Figure 2, g_2 is the only one for which $BC \to A$ holds. The tuples t_1, t_4 represent a counter-example to $BC \to A$ for g_1 and g_3. Realities supply an abstract support to practitioners to help them understand under which conditions their background knowledge, modeled as a functional dependency, turns out to be indeed true or false in their data.

3 Preliminaries

In this paper, we only consider finite structures. First, we recall basic definitions about relational databases [Mai83, MR92]. Then, we move to lattice theoretical concepts [DP02, Grä11].

Relational model A *relation scheme* R is a (finite) set of *attributes* $A_1, \ldots, A_n$, for $n \in \mathbb{N}$. Its powerset is written 2^R. Each attribute A_i has values in some set $\mathsf{dom}(A_i)$

called the *domain* of A_i. We consider a unique null value, `null`, which can be part of every attribute domain. The domain of R is the union of attributes domains, that is $\mathsf{dom}(R) = \bigcup_{A \in R} \mathsf{dom}(A)$. Capital first letters of the alphabet A, B, C represent attributes, while capital last letters such as X, Y, Z denote sets of attributes. We write XY as a shortcut for $X \cup Y$, and $A_1A_2A_3$ for $\{A_1, A_2, A_3\}$. A *tuple* t over R is a mapping $t \colon R \to \mathsf{dom}(R)$ such that $t[A] \in \mathsf{dom}(A)$ for each $A \in R$, where $t[A]$ is the projection of t on A. Similarly, $t[X]$, for some $X \subseteq R$, denotes the projection of t on X. A *relation* r is a collection of tuples over R. A *functional dependency* (FD) over R is a pair (X, A) usually written $X \to A$, where $X \subseteq R$ and $A \in R$. Without loss of generality, we assume that $X \to A$ is not trivial, i.e., $A \notin X$. Let r be a relation over R. We say that r *satisfies* the functional dependency $X \to A$ if for each pair t_1, t_2 of tuples in r, $t_1[X] = t_2[X]$ implies $t_1[A] = t_2[A]$. This is written $r \models X \to A$. If $\mathbf{F}$ is a set of functional dependencies, we write $r \models \mathbf{F}$ to denote that $r \models X \to A$ for each $X \to A \in \mathbf{F}$.

Lattices, implications Let S be a set. A *partial order* or *poset* is a pair $(S, \leq)$, where $\leq$ is a binary relation that is reflexive, transitive and anti-symmetric. Let $x, y \in S$. We say that x and y are *comparable* if $x \leq y$ or $y \leq x$. Otherwise, they are *incomparable*. A *lower bound* ℓ of x and y is an element of S satisfying $\ell \leq x$ and $\ell \leq y$. Moreover, if for any lower bound ℓ' of x and y we have $\ell' \leq \ell$, ℓ is called the *meet* of x and y. We denote it by $x \wedge y$. Dually, an *upper bound* of x and y is an element $u \in S$ such that $x \leq u$ and $y \leq u$. If u is minimal with respect to all upper bounds of x and y, it is the *join* of x and y. We write it $x \vee y$. For $X \subseteq S$, the meet $\bigwedge X$ and join $\bigvee X$ of X are defined accordingly. A poset in which every pair of elements has both a meet and a join is a *lattice*. A lattice will be noted $\mathcal{L}(S, \leq, \vee, \wedge)$ or simply $\mathcal{L}$ when precision is not necessary. It has a minimum element 0 called the *bottom*, and a maximum one, the *top*, denoted by 1. Let $\mathcal{L}$ be a lattice, and $x, y, z \in \mathcal{L}$ such that $y < x$ (i.e., $y \leq x$ and $y \neq x$). If for any $z \in \mathcal{L}$, $y < z \leq x$ implies $x = z$, then we say that x *covers* y and we write $y \prec x$. An element $a \in \mathcal{L}$ which covers the bottom 0 of $\mathcal{L}$ is called an *atom*. We denote by $\mathcal{A}(\mathcal{L})$ the set of atoms of $\mathcal{L}$. Let $m \in \mathcal{L}$. We say that m is *meet-irreducible* if for any $x, y \in \mathcal{L}$, $m = x \wedge y$ implies $m = x$ or $m = y$. A *join-irreducible* element j is defined dually with $\vee$. Meet-irreducible elements (resp. join-irreducible elements) of $\mathcal{L}$ are denoted by $\mathcal{M}(\mathcal{L})$ (resp. $\mathcal{J}(\mathcal{L})$). Let $\mathcal{L}_1, \mathcal{L}_2$ be two lattices. A map $\varphi \colon \mathcal{L}_1 \to \mathcal{L}_2$ is *increasing* if for any $x, y \in \mathcal{L}_1$, $x \leq y$ implies $\varphi(x) \leq \varphi(y)$. It is a $\wedge$-*homomorphism* if for any $x, y \in \mathcal{L}_1$, $\varphi(x \wedge y) = \varphi(x) \wedge \varphi(y)$. *Decreasing* maps and $\vee$-*homomorphisms* are defined dually. The map φ is a *homomorphism* if it is both a $\vee$-homomorphism and a $\wedge$-homomorphism. If $\mathcal{L}_1, \ldots, \mathcal{L}_n$ are lattices, the cartesian product $\prod_{i=1}^{n} \mathcal{L}_i$ with component-wise induced order is the *direct product* of $\mathcal{L}_1, \ldots, \mathcal{L}_n$. We have $x \leq y$ if and only if $x_i \leq y_i$ for every $1 \leq i \leq n$. If $\mathcal{L}$ is a lattice, a $\wedge$-*sublattice* $\mathcal{L}'$ of $\mathcal{L}$ is a lattice which is a subset of $\mathcal{L}$ such that $x, y \in \mathcal{L}'$ implies $x \wedge y \in \mathcal{L}'$.

Now, we introduce lattice implications and closure operators, based on definitions and results of [Day92]. Let $\mathcal{L}$ be a lattice and $\phi \colon \mathcal{L} \to \mathcal{L}$. The map ϕ is a *closure operator* if for any $x, y \in \mathcal{L}$, $x \leq \phi(x)$ (extensive), $x \leq y \implies \phi(x) \leq \phi(y)$ (increasing) and $\phi(\phi(x)) = \phi(x)$ (idempotent). Then, $\phi(x)$, is the *closure* of x with respect to ϕ. The set $\mathcal{L}_\phi = \phi(\mathcal{L}) = \{x \in \mathcal{L} \mid \phi(x) = x\}$ of fixed points (or closed elements) of ϕ is a $\wedge$-sublattice of $\mathcal{L}$ which contains the top of $\mathcal{L}$. Therefore, we also obtain $\phi(x) = \bigwedge\{y \in \mathcal{L}_\phi \mid x \leq y\}$. The set of all such $\wedge$-sublattices of $\mathcal{L}$ is denoted $\mathsf{Sub}^1_\wedge(\mathcal{L})$. When $\mathcal{L}$ equals 2^R for some attribute set R, and ϕ is a closure operator on $\mathcal{L}$, $\mathcal{L}_\phi$ is a *closure system* [Grä11] (in this case $\wedge$ is the set intersection $\cap$). A *(lattice) implication* is a pair $x \to y$ such that $x, y \in \mathcal{L}$.

We say that an element $z \in \mathcal{L}$ satisfies the implication $x \to y$ if $x \leq z \implies y \leq z$, and we write $z \models x \to y$. Let Σ be a set of implications over $\mathcal{L}$. We write $z \models \Sigma$ if $z \models x \to y$ for each $x \to y \in \Sigma$. We also say that z is a *model* of Σ. Moreover, Σ induces a closure operator ϕ^Σ on $\mathcal{L}$ such that $x \in \mathcal{L}$ is closed if and only if $x \models \Sigma$. Formally, $\mathcal{L}_{\phi^\Sigma} = \{x \in \mathcal{L} \mid x \models \Sigma\}$. Dually, any closure operator ϕ on $\mathcal{L}$, and hence any $\mathcal{L}' \in \mathsf{Sub}^1_\wedge(\mathcal{L})$, can be represented by implications [Day92]. Let $\mathcal{L}' \in \mathsf{Sub}^1_\wedge(\mathcal{L})$ and $x \to y$ be an implication on $\mathcal{L}$. We obtain $\mathcal{L}'$ *satisfies* $x \to y$, written $\mathcal{L}' \models x \to y$ if for any $z \in \mathcal{L}'$, $z \models x \to y$. The notion $\mathcal{L}' \models \Sigma$ follows. We have the following folklore property: $\mathcal{L}' \models x \to y$ if and only if $y \leq \phi(x)$.

Example 6. Let us first consider the lattice $\mathcal{L}$, to the left of Figure 4. The atoms are a, b and c. Now consider the set $\Sigma = \{e \to d, b \to d\}$ of lattice implications. The corresponding $\wedge$-sublattice $\mathcal{L}_{\phi^\Sigma}$ of $\mathcal{L}$ is represented to the right of Figure 4. For instance, f is not closed with respect to Σ because it does not satisfy the implication $b \to d$: $b \leq f$ but $d \not\leq f$. The lattice $\mathcal{L}_{\phi^\Sigma}$ is also associated to a closure operator ϕ_Σ mapping an element of $\mathcal{L}$ to an element $\mathcal{L}_{\phi^\Sigma}$. For example, the closure $\phi_\Sigma(b)$ of b is d: it is the unique minimum element x of $\mathcal{L}_{\phi^\Sigma}$ satisfying $b \leq x$.

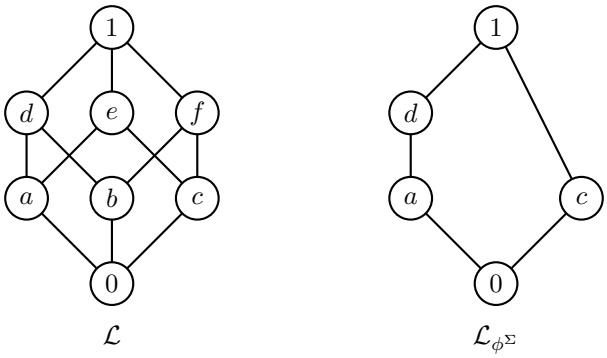


Figure 4: The two lattices of Example 6

4 A lattice-based framework to handle comparabilities

In this section, we formally define our framework. Then, we show the relationships between the classical relational model and our framework.

4.1 Defining the framework

Let R be a relation scheme. For every attribute $A \in R$, the first step is to define abstract values whenever two elements of $\mathsf{dom}(A)$ are compared. The *abstract domain* of A, denoted by $\mathsf{dom}_{abs}(A)$, is a finite set of *abstract values* disjoint from the domain $\mathsf{dom}(A)$, representing comparabilities between every pair of values in $\mathsf{dom}(A)$. We assume that a partial order $\leq_A$, or simply $\leq$ when no confusion arises, exists over $\mathsf{dom}_{abs}(A)$ such that the poset $\mathcal{L}_A = (\mathsf{dom}_{abs}(A), \leq_A)$ is a lattice. We say that $\mathcal{L}_A$ is the *abstract lattice* of A. We put 0_A and 1_A as the bottom and top elements (resp.) of $\mathcal{L}_A$. The *abstract domain* $\mathsf{dom}_{abs}(R)$ of R is the direct product of abstract domains, i.e., $\mathsf{dom}_{abs}(R) = \prod_{A \in R} \mathsf{dom}_{abs}(A)$. Two attributes can have the same abstract domain.

Now, we define the explicit mapping of pairs of domain values to abstract values through a comparability function. Let A be an attribute in R. A *comparability function* for A is a

surjective map f_A from $\mathsf{dom}(A) \times \mathsf{dom}(A)$ to $\mathsf{dom}_{abs}(A)$ which is commutative, i.e., such that $f_A(x_A, y_A) = f_A(y_A, x_A)$ for all $x_A, y_A \in \mathsf{dom}(A)$.

Remark 1. When using the framework, one can further require reflexivity on the comparability functions, i.e. $f(x_A, x_A) = 1_A$, in order to get a semantic of comparability closer to equality. Even more, it could be possible to make the functions reflexive on all values but `null` where some freedom is allowed. Indeed, in practice the meaning of the `null` value in the data should be explained by domain experts, along with recommendations on how to deal with it. Moreover, since the `null` value indicates a missing value, relaxing reflexivity of comparability functions on `null` allows to consider absent values as possibly different. Interestingly, while the results we present in the body of this paper are unchanged by reflexivity, we show in C that reflexivity is a key property to ensure completeness of (extended) Armstrong axioms.

Our approach ensures that the data remains unchanged, and bypasses usual ways of dealing with incomplete data such as completion and repairing [AL18, Ber11, Lib16].

Let t_1, t_2 be two tuples over R. The comparability function f_R for R is the combination of comparability functions on attributes of R. More precisely, for every two tuples t_1, t_2 over R, we have:

$$f_R(t_1, t_2) = \langle f_{A_1}(t_1[A_1], t_2[A_1]), \ldots, f_{A_n}(t_1[A_n], t_2[A_n]) \rangle$$

Note that $f_R(t_1, t_2)$ is a generalization of agree sets between two tuples [BDFS84]. For any relation r over R, we put $f_R(r) = \{f_R(t_1, t_2) \mid t_1, t_2 \in r\}$. Let us denote by $\mathcal{L}_R = (\mathsf{dom}_{abs}(R), \leq_R)$ the direct product of abstract lattices. An element of $\mathsf{dom}_{abs}(R)$ is an *abstract tuple*. It is a vector $\langle a_1, \ldots, a_n \rangle$ where $a_i \in \mathsf{dom}_{abs}(A_i)$ for every $1 \leq i \leq n$. We can use $x, y \in \mathcal{L}_R$ as a shortcut for $x, y \in \mathsf{dom}_{abs}(R)$. Also, we will make a slight abuse of the [ ] notation: if $x \in \mathcal{L}_R$ we denote by $x[A]$ the projection of the vector x on A. We are now ready to introduce attribute and scheme contexts:

Definition 1 (attribute context, scheme context)**.** Let R be a relation scheme, and let $A \in R$. An *attribute context* for A is a triple $\mathcal{C}_A = (A, f_A, \mathcal{L}_A)$ where $\mathcal{L}_A = (\mathsf{dom}_{abs}(A), \leq_A)$ is an abstract lattice and $f_A \colon \mathsf{dom}(A) \times \mathsf{dom}(A) \to \mathsf{dom}_{abs}(A)$ a comparability function. A *scheme context* for $\mathcal{C}_R$ is a union of attribute contexts for each $A \in R$, i.e., $\mathcal{C}_R = \{\mathcal{C}_A \mid A \in R\}$.

Complexity assumption: for all the computational results of this paper, the contexts are given as part of the input. For a given attribute context $\mathcal{C}_A = (A, f_A, \mathcal{L}_A)$, we assume that f_A can be computed in polynomial time in the size of its input. The lattice $\mathcal{L}_A$ is given in such a way that the operations $\leq$, $\vee$, $\wedge$, can be conducted in polynomial time in the size of $\mathcal{L}_A$.

From a scheme context $\mathcal{C}_R$, we have both its associated comparability function f_R and its abstract domain $\mathsf{dom}_{abs}(R)$, along with the lattice $\mathcal{L}_R$. If no confusion is possible, the subscript R will be omitted, i.e., we will use $\leq, \wedge, \vee$ instead of $\leq_R, \wedge_R, \vee_R$, respectively. In the sequel, a relation r will be defined over a scheme context $\mathcal{C}_R$, instead of R.

Example 7. In the running example, the abstract domain of A is

$$\mathsf{dom}_{abs}(A) = \{\textit{equal}, \textit{distributed}, \textit{similar}, \textit{abnormal}\},$$

and the corresponding abstract lattice is $\mathcal{L}_A$, illustrated in Figure 1. The attribute context of A is $(A, f_A, \mathcal{L}_A)$. The scheme context of R is then $\mathcal{C}_R = \{(A, f_A, \mathcal{L}_A), (B, f_B, \mathcal{L}_B),$

$(C, f_C, \mathcal{L}_C)\}$. For instance, the abstract tuple $\langle s, d_B, c\rangle$ results from the comparison of t_3, t_4 in the relation Patients: $f_R(t_3, t_4) = \langle f_A(3.2, 3.5), f_B(M, F), f_C(72, 73)\rangle = \langle s, d_B, c\rangle$. In Table 2, we give (a part of) the family f_R(Patients) of abstract tuples associated with the relation Patients.

f_R(Patients)	A	B	C
t_1, t_1	e	e	c
t_1, t_2	e	e	u
t_1, t_3	d_A	d_B	c
t_1, t_4	d_A	e	c
$\dots$	$\dots$	$\dots$	$\dots$
t_5, t_5	e	e	c

Table 2: The abstract tuples associated to the relation Patients

Next, we define (attribute) interpretations as particular mappings from an abstract lattice to $\{0, 1\}$. Their aim is to decide for each two elements in the attribute domain whether they can be considered equal.

Definition 2 (interpretation)**.** Let R be a relation scheme, $A \in R$ and $\mathcal{C}_A = (A, f_A, \mathcal{L}_A)$ an attribute context for A. An *attribute (context) interpretation* for A is a map $h_A \colon \mathcal{L}_A \to \{0, 1\}$ satisfying the following properties:

1. $x \leq y$ in $\mathcal{L}_A$ implies $h_A(x) \leq h_A(y)$, i.e., h_A is increasing,
2. $h_A(1_A) \neq h_A(0_A)$.

In particular, an interpretation must be surjective, since $h_A(1_A) = 1$ and $h_A(0_A) = 0$. Practically, it guarantees that two values in $\mathsf{dom}(A)$ can always be interpreted as equal or different. Moreover, the interpretation has to be increasing. Intuitively, if an abstract value x_A of $\mathcal{L}_A$ is interpreted as 1 (i.e., equality) by h_A, any value $y_A \geq_A x_A$ must be set to 1 since it is closer to *"true"* equality than x_A.

Definition 3 (scheme interpretation)**.** Let $R = \{A_1, \dots, A_n\}$ be a relation scheme, and $\mathcal{C}_R$ a scheme context. Let h_{A_i} be an attribute interpretation for A_i, for all $1 \leq i \leq n$. The map $g \colon \mathcal{L}_R \to \{0, 1\}^n$ defined by $(g(x))[A_i] = h_{A_i}(x[A_i])$, for each $x \in \mathcal{L}_R$ and each $1 \leq i \leq n$, is a *scheme (context) interpretation*. In the sequel, $g_{|A_i}$ will denote the restriction of g to the attribute $A_i \in R$, i.e., $g_{|A_i} = h_{A_i}$.

A scheme interpretation g maps each possible abstract tuple to a binary vector in $\{0, 1\}^n$. This binary vector can be seen as the characteristic vector of some subset of R. In the sequel, we will use $g(\cdot)$ interchangeably to denote a set or its characteristic vector.

Example 8. We continue our running example. In Figure 2, we give three possible scheme interpretations. Above the dotted lines (white nodes), the abstract values are interpreted as 1, and 0 otherwise (grey nodes). For instance, the abstract tuple $\langle e, d_B, u\rangle$ is interpreted as $\langle 1, 0, 0\rangle$ with g_1 and g_2. It is interpreted as $\langle 1, 0, 1\rangle$ with g_3. In Table 3, we give (some of) the interpretations of the comparison of the tuples of Patients. For example, $g_1(f_R(t_1, t_2)) = g_1(\langle e, e, u\rangle) = \langle 1, 1, 0\rangle$ and $g_3(f_R(t_1, t_2)) = \langle 1, 1, 1\rangle$. Thus, t_1 and t_2 are considered equal under the interpretation g_3 but different for g_1 and g_2.

	$g_1(\cdot)$	$g_2(\cdot)$	$g_3(\cdot)$
t_1, t_1	$\langle 1,1,1\rangle$	$\langle 1,1,1\rangle$	$\langle 1,1,1\rangle$
t_1, t_2	$\langle 1,1,0\rangle$	$\langle 1,1,0\rangle$	$\langle 1,1,1\rangle$
t_1, t_3	$\langle 0,0,1\rangle$	$\langle 1,0,1\rangle$	$\langle 0,0,1\rangle$
t_1, t_4	$\langle 0,1,1\rangle$	$\langle 1,1,1\rangle$	$\langle 0,1,1\rangle$
$\dots$	$\dots$	$\dots$	$\dots$
t_5, t_5	$\langle 1,1,1\rangle$	$\langle 1,1,1\rangle$	$\langle 1,1,1\rangle$

Table 3: Comparing tuples of Patients with the interpretations of Figure 2

The following statement means that a scheme interpretation g is uniquely defined by its components.

Proposition 1. *Let g be a scheme interpretation. The following properties hold:*

1. *g increasing if and only if h_{A_i} is increasing for any $1 \leq i \leq n$,*
2. *g is a $\wedge$-homomorphism (resp. $\vee$-homomorphism) if and only if for any $1 \leq i \leq n$, h_{A_i} is a $\wedge$-homomorphism (resp. $\vee$-homomorphism) .*

4.2 Applications to the classical relational model

In this part, we show how to express the classical relational model in our framework. We also study the scenario of SQL's 3-valued logic including `null` values.

Relational model (without nulls) Given a relation scheme $R = \{A_1, \dots, A_n\}$, we associate to each $A_i \in R$ the attribute context $(A_i, f_{A_i}, \mathcal{L}_{A_i})$ where $\mathcal{L}_{A_i} = (\{0,1\}, \leq)$ and f_{A_i} is the strict equality, i.e., $f_{A_i}(x,x) = 1$ and 0 otherwise. Thus, the abstract tuples associated with a relation are binary vectors over $\{0,1\}^n$. In other words, they are the agree sets of the input relation [BDFS84]. The context $\mathcal{C}_R = \{(A_i, f_{A_i}, \mathcal{L}_{A_i}) \mid A_i \in R\}$ has a unique interpretation g such that $g_{|A_i}(x[A_i]) = 1$ if $x[A_i] = 1$ and 0 otherwise, for each $A_i \in R$. The composition $g(f_R)$ precisely depicts the equality between two tuples. Moreover, this context suggests that in practice, a domain expert can leave classical equality as a default comparability and apply dedicated comparability functions only on some relevant attributes. In this particular setup, possible and certain FDs (see Section 5) coincide, and they are precisely the valid classical FDs. Moreover, abstract functional dependencies (see Section 5) are the expression of classical FDs in terms of characteristic vectors. As a consequence, an abstract FD is valid if and only if the corresponding FD is true in the relation. It follows that abstract FDs can be seen as a generalization of classical FDs. As such, the hardness of problems on abstract FDs, such as identifying a cover of abstract FDs for a relation [Mai83], inherits the complexity of their counterpart in terms of classical FDs. The reader can refer to C for more details on computational aspects of abstract FDs.

SQL model with nulls A similar approach can be applied to model SQL's three-valued logic with `nulls`. Namely, if $R = \{A_1, \dots, A_n\}$ is a relation scheme, we assign to each attribute A_i of R the following attribute context $(A_i, f_{A_i}, \mathcal{L}_{A_i})$:

- $\mathcal{L}_{A_i} = (\{0, u, 1\}, \leq)$ and $\leq$ is defined by $0 \leq u \leq 1$ (u stands for *unknown*),

- $f_{A_i}(x, x) = 1$ if $x \neq \texttt{null}$, $f_{A_i}(x, y) = u$ if x or y is $\texttt{null}$ and 0 otherwise.

Usually, functional dependencies are not defined in the presence of `null` values. With the help of interpretations, our framework offers an easy way to tell whether two tuples are equal in the presence of `null`. Even more, it proposes two types of functional dependencies: (i) abstract FDs, using the *unknown* value in abstract tuples, and (ii) FDs based on interpretations of the context (see Section 5 for formal a definitions of abstract FDs).

5 Application to functional dependencies

In this section, we study functional dependencies in light of our framework. First, we show that each relation is associated with an *abstract lattice*, which can be represented by lattice implications called *abstract functional dependencies*.

Second, we characterize those interpretations that guarantee that any abstract lattice is turned into a closure system, thus paving the way for functional dependencies [DLM92]. These interpretations are called *(strong) realities*. Using realities, we establish several results connecting AFDs and classical FDs. Complementary results regarding abstract FDs can be found in C.

In the third part of this section, we use realities to introduce *possible* and *certain* functional dependencies. A FD is (strongly) possible if there exists a (strong) reality for which it holds, and (strongly) certain if it holds for every (strong) reality. In particular, we show that the decision as to whether a given FD is possible/certain can be conducted in polynomial time. We prove however that deciding strong possibility is NP-complete.

5.1 Abstract functional dependencies, realities

Let r be a relation over a scheme context $\mathcal{C}_R$. As a reminder, $\mathcal{L}_R$ is the product of abstract lattices $\prod_{i=1}^{n} \mathcal{L}_{A_i}$. The *abstract lattice* associated with r is

$$\mathcal{L}_r = \left(\left\{ \bigwedge T \mid T \subseteq f_R(r) \right\}, \leq_R \right)$$

As usual, $\bigwedge \emptyset$ equals the top element of $\mathcal{L}_R$ [DP02, Day92]. Observe that $\mathcal{L}_r$ is a subset of $\mathcal{L}_R$. It is in fact a $\wedge$-sublattice of $\mathcal{L}_R$.

Proposition 2. *Let r be a relation over a scheme context $\mathcal{C}_R$. Then, $\mathcal{L}_r \in \mathsf{Sub}^1_\wedge(\mathcal{L}_R)$.*

Example 9. We continue our running example. The lattice $\mathcal{L}_R$ is given in Figure 5. We illustrate the abstract lattice associated to the relation Patients in Figure 6. For instance, the abstract tuple $\langle s, d_B, u \rangle$ is obtained by taking $\langle s, d_B, c \rangle \wedge \langle e, e, u \rangle = f_R(t_3, t_4) \wedge f_R(t_1, t_2)$.

Since $\mathcal{L}_r$ is a $\wedge$-sublattice of $\mathcal{L}_R$, it corresponds to a closure operator ϕ. Following [Day92], $\mathcal{L}_r$ can be represented by a set Σ of lattice implications, the so-called *abstract functional dependencies* in our framework.

Definition 4 (Syntax)**.** Let R be a relation scheme, $\mathcal{C}_R$ a scheme context, and $\mathsf{dom}_{abs}(R)$ its abstract domain. An *abstract functional dependency* over $\mathcal{C}_R$ is an expression of the form $x \to y$ where $x, y \in \mathsf{dom}_{abs}(R)$.

Definition 5 (Satisfaction)**.** Let r be a relation over $\mathcal{C}_R$ and $x, y \in \mathsf{dom}_{abs}(R)$. We say that r satisfies $x \to y$ with respect to $\mathcal{C}_R$, denoted by $r \models_{\mathcal{C}_R} x \to y$, if for all $t_1, t_2 \in r$, $x \leq f_R(t_1, t_2)$ entails $y \leq f_R(t_1, t_2)$.

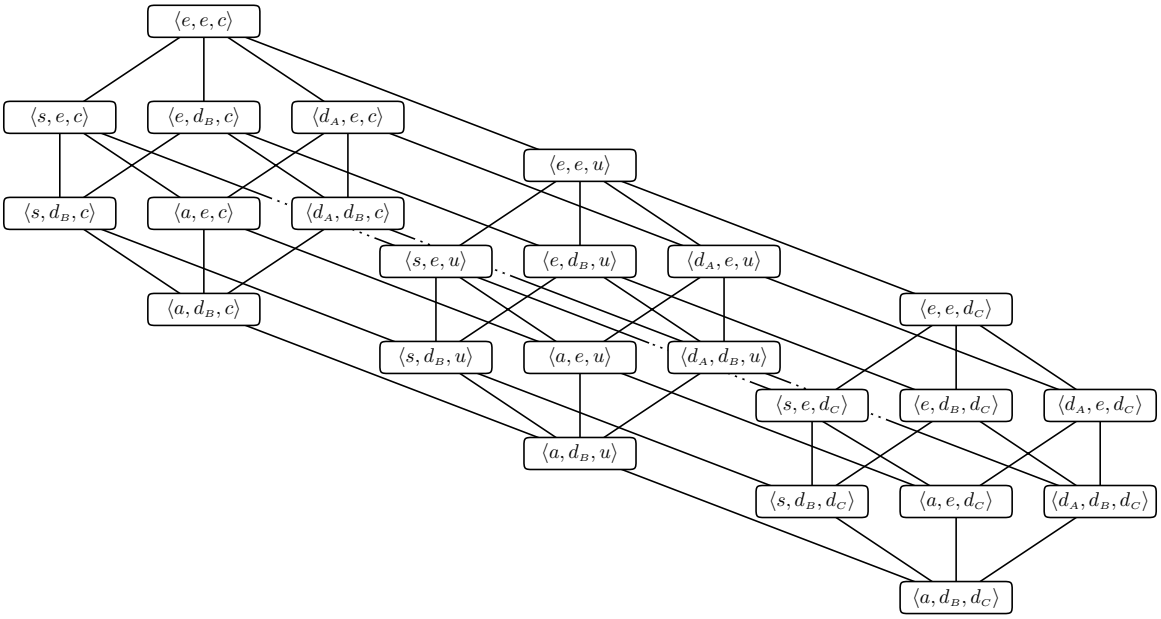


Figure 5: The product $\mathcal{L}_R$ of abstract lattices of the context $\mathcal{C}_R$

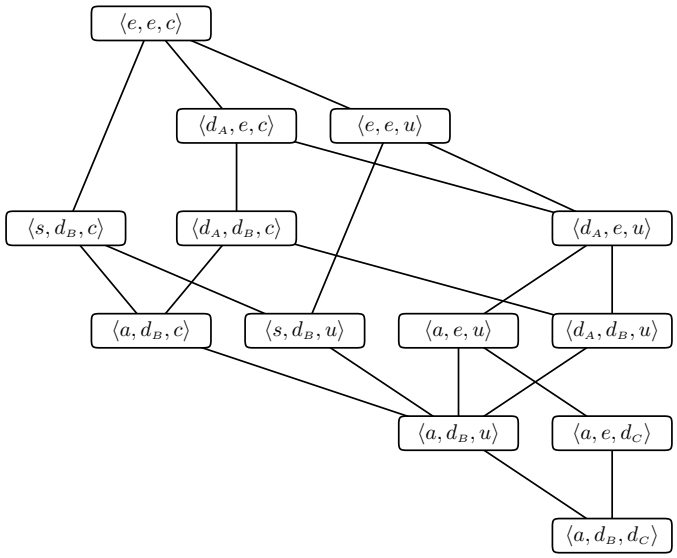


Figure 6: The abstract lattice associated with the relation Patients.

When no confusion is possible, we will drop the element $\mathcal{C}_R$ of notations regarding functional dependencies. Namely, we will denote by $x \rightarrow y$ an abstract functional dependency over $\mathcal{C}_R$ and $r \models x \rightarrow y$ when r models $x \rightarrow y$ with respect to $\mathcal{C}_R$. We first establish a natural relationship between the satisfaction of an abstract FD in a relation and its validity in the corresponding abstract lattice.

Proposition 3. *Let r be a relation over $\mathcal{C}_R$, and $x \rightarrow y$ an abstract functional dependency. Then $r \models x \rightarrow y$ if and only if $\mathcal{L}_r \models x \rightarrow y$.*

Remark 2. Since $f_R(r)$ is a generating set of $\mathcal{L}_r$, and for each $x \in \mathcal{L}_r$ we have $\phi(x) = \bigwedge\{y \in \mathcal{L}_r \mid x \leq y\} = \bigwedge\{y \in f_R(r) \mid x \leq y\}$, it is possible to compute $\phi(x)$ in polynomial time in the size of r and $\mathcal{C}_R$ (i.e., if the abstract lattice of each attribute is given). Since an abstract FD $x \rightarrow y$ holds in $\mathcal{L}_r$ if and only if $y \leq \phi(x)$, testing the validity of an abstract FD can also be conducted in polynomial time.

r	T. level (A)	Gender (B)	W. size (C)
t_1	1.4	M	`null`
t_2	3.2	F	10
t_3	3.5	F	73

Table 4: The relation r

Abstract functional dependencies depict dependencies between the attributes of a relation in the absence of interpretations. More precisely, the meaning of an abstract FD $x \rightarrow y$ valid in a relation r is *"the similarity of two tuples is greater than y if it is greater than x"*.

Example 10. We continue our running example. Let us first consider the abstract FD $\langle e, d_B, d_C\rangle \rightarrow \langle s, e, u\rangle$. To determine whether it holds in Patients, we show that $\langle s, e, u\rangle \leq \phi(\langle e, d_B, d_C\rangle)$ where $\phi(\langle e, d_B, d_C\rangle)$ is the closure of $\langle e, d_B, d_C\rangle$ in the associated abstract lattice (see Figure 6). We have $\phi(\langle e, d_B, d_C\rangle) = \bigwedge\{x \in f_R(\text{Patients}) \mid \langle e, d_B, d_C\rangle \leq x\} = \langle e, e, u\rangle$. Then, $\langle s, e, u\rangle \leq \langle e, e, u\rangle$ and $\langle e, d_B, d_C\rangle \rightarrow \langle s, e, u\rangle$ is a valid abstract FD. Dually, consider the abstract FD $\langle a, e, c\rangle \rightarrow \langle e, d_B, d_C\rangle$. There exists x in the lattice such that $\langle a, e, c\rangle \leq x$ but $\langle e, d_B, c\rangle \not\leq x$ (take $x = \langle d_A, e, c\rangle$). Therefore, Patients $\models \langle a, e, c\rangle \rightarrow \langle e, d_B, c\rangle$ does not hold.

Now, we examine the connection between abstract functional dependencies and functional dependencies. Let r be a relation over a scheme context $\mathcal{C}_R$. Its associated abstract lattice is $\mathcal{L}_r$. An interpretation g over $\mathcal{C}_R$ maps $\mathcal{L}_r$ to the set system $g(\mathcal{L}_r) = \{g(x) \mid x \in \mathcal{L}_r\}$ over R. We illustrate this interpretation in the next example.

Example 11. We consider the scheme context of our running example. Let r be the relation presented in Table 4. In Figure 7, we give the abstract lattice $\mathcal{L}_r$ and its interpretation $g_1(\mathcal{L}_r)$ using g_1. Observe that $g_1(\mathcal{L}_r)$ is a closure system. Hence, it can be represented by a set of functional dependencies. In particular, it satisfies $C \rightarrow A$ and $A \rightarrow B$.

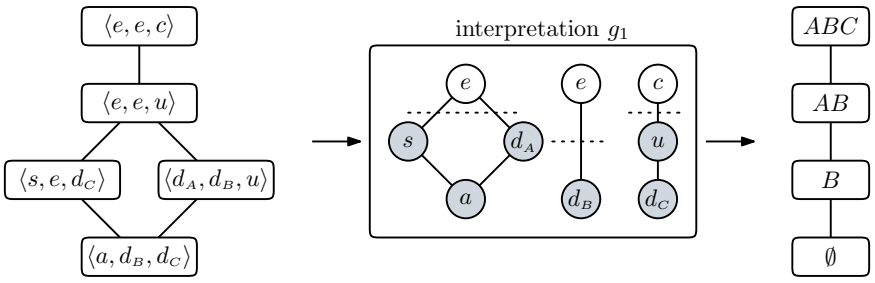


Figure 7: $\mathcal{L}_r$ and its interpretation $g_1(\mathcal{L}_r)$ using g_1

Thus, the interpretation of an abstract lattice may be a closure system, which can be represented by functional dependencies. However, not all interpretations enjoy this property, as shown by the following example.

Example 12. We continue the previous example. Instead of g_1, we consider the interpretation g_1' depicted in Figure 8. The resulting set system $g_1'(\mathcal{L}_r)$ is not closed under intersection. Henceforth, it is not a closure system, and it cannot be represented by functional dependencies.

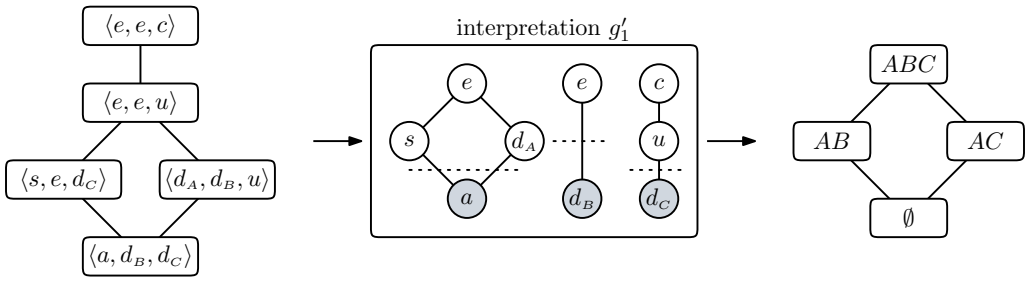


Figure 8: $\mathcal{L}_r$ and its interpretation $g'_1(\mathcal{L}_r)$ using g'_1

Thus, in view of functional dependencies, the following question arises: which property should a scheme interpretation have in order to guarantee that the interpretation of any abstract lattice is a closure system? This question is central since functional dependencies must satisfy Armstrong axioms, and their model must form a closure system [DLM92]. First, we answer the case where R has few attributes.

Proposition 4. *Let $\mathcal{C}_R$ be a context scheme with $R = \{A_1, \ldots, A_n\}$. If $n \leq 2$, then for any scheme interpretation g and any $\mathcal{L} \in \mathsf{Sub}^1_\wedge(\mathcal{L}_R)$, $g(\mathcal{L})$ is a closure system over R.*

Proof. If $n = 2$, then for any $\mathcal{L} \in \mathsf{Sub}^1_\wedge(\mathcal{L}_R)$, $g(\mathcal{L})$ is a non-empty subset of $\{0,1\}^2$ containing $\langle 1,1\rangle$ by definition of $\mathsf{Sub}^1_\wedge(\mathcal{L}_R)$. The only subset of $\{0,1\}^2$ possessing $\langle 1,1\rangle$ which is not closed with respect to $\wedge$ is $\{\langle 0,1\rangle, \langle 1,0\rangle, \langle 1,1\rangle\}$. However, if $g(\mathcal{L})$ contains $\langle 0,1\rangle$ and $\langle 1,0\rangle$, there exists $x, y \in \mathcal{L}$ such that $g(x) = \langle 0,1\rangle$ and $g(y) = \langle 1,0\rangle$. As $\mathcal{L}$ is a $\wedge$-sublattice of $\mathcal{L}_R$, and g is increasing, it must be that $x \wedge y \in \mathcal{L}$, $g(x \wedge y) \leq g(x)$ and $g(x \wedge y) \leq g(y)$ so that necessarily $g(x \wedge y) = \langle 0,0\rangle$. Therefore, $g(\mathcal{L})$ is indeed a closure system, concluding the proof. □

In the more general case where the number of attributes is unbounded, this property may not hold, as shown by the previous example. In the next theorem, we characterize such interpretations using lattice homomorphisms.

Theorem 1. *Let $\mathcal{C}_R$ be a context scheme with $R = \{A_1, \ldots, A_n\}$, $n \geq 3$, and let g be a scheme interpretation. Then g is a $\wedge$-homomorphism if and only if for any $\mathcal{L} \in \mathsf{Sub}^1_\wedge(\mathcal{L}_R)$, $g(\mathcal{L})$ is a closure system over R.*

Proof. We begin with the only if part. Let g be a $\wedge$-homomorphism and $\mathcal{L} \in \mathsf{Sub}^1_\wedge(\mathcal{L}_R)$. Consider $x, y \in \mathcal{L}$ such that $g(x)$ and $g(y)$ are incomparable. Then, since $\mathcal{L} \in \mathsf{Sub}^1_\wedge(\mathcal{L}_R)$ and g is a $\wedge$-homomorphism, it follows that $x \wedge y \in \mathcal{L}$ and $g(x \wedge y) = g(x) \wedge g(y) \in g(\mathcal{L})$. Furthermore, the top element $\langle 1_1, \ldots, 1_n\rangle$ belongs to $\mathcal{L}$ by assumption and $g(\langle 1_1, \ldots, 1_n\rangle) = R \in g(\mathcal{L})$ by definition of g. Hence $g(\mathcal{L})$ is a closure system over R.

For the if part, we use contrapositive. Suppose g is not a $\wedge$-homomorphism. We construct $\mathcal{L} \in \mathsf{Sub}^1_\wedge(\mathcal{L}_R)$ such that $g(\mathcal{L})$ is not a closure system. Recall by Proposition 1 that if g is not a $\wedge$-homomorphism, then it must be that at least one of the h_A, $A \in R$ is not a $\wedge$-homomorphism either. Without loss of generality, assume that $A = A_1$. Since h_{A_1} is increasing but not $\wedge$-homomorphic, there must be $x_1, y_1 \in \mathcal{L}_{A_1}$ such $h_{A_1}(x_1) = h_{A_1}(y_1) = 1$ and $h_{A_1}(x_1 \wedge y_1) = 0$. Recall that we assumed $n \geq 3$. Let $\mathcal{L}$ be the four element set:

$$\{\langle x_1, 1_2, 0_3, 1_4, \ldots, 1_n\rangle, \langle y_1, 0_2, 1_3, 1_4, \ldots, 1_n\rangle, \langle x_1 \wedge y_1, 0_2, 0_3, 1_4, \ldots, 1_n\rangle, \langle 1_1, 1_2, 1_3, 1_4, \ldots, 1_n\rangle\}$$

Note that $\mathcal{L}$ is a $\wedge$-sublattice of $\mathcal{L}_R$ which contains its top element. Since g is increasing and we constrained $h_{A_i}(0_{A_i}) = 0$ and $h_{A_i}(1_{A_i}) = 1$, we obtain:

$$g(\mathcal{L}) = \{\langle 1,1,0,1,\ldots,1\rangle, \langle 1,0,1,1,\ldots,1\rangle, \langle 0,0,0,1,\ldots,1\rangle, \langle 1,1,1,1,\ldots,1\rangle\}$$

which is not a closure system when interpreted as attribute sets, thus concluding the proof. □

For each attribute, the $\wedge$-homomorphism property defines a minimum element in the abstract lattice which is interpreted as 1. Still, $\wedge$-homomorphisms only partially capture the intuitive behavior that an interpretation could have. Indeed, while the meet of two abstract values set to 1 equals 1 by $\wedge$-preservation of g, the join of two abstract values interpreted as 0 could be 1. Hence, we are led to consider two types of interpretations. First, $\wedge$-homomorphisms, which guarantee the semantic for functional dependencies thanks to Theorem 1. Second, homomorphisms, a strengthening of $\wedge$-homomorphisms. These types of interpretations are called realities and strong realities, respectively.

Definition 6 (reality, strong reality)**.** Let $\mathcal{C}_R$ be a scheme context. A scheme interpretation g is a *reality* if it is a $\wedge$-homomorphism over $\mathcal{L}_R$. It is a *strong reality* if it is a homomorphism.

We denote by $\mathcal{R}$ the set of realities of $\mathcal{L}_R$. The set of strong realities is $\mathcal{R}_s$. For a given reality g, let x_g be the abstract tuple of $\mathcal{L}_R$ satisfying $x_g[A] = \mathsf{min}_{\leq}\{x_A \in \mathcal{L}_A \mid g_{|A}(x_A) = 1\}$ for every attribute A. As g is an increasing $\wedge$-homomorphism, such an x_g is uniquely defined. Moreover, $x_g[A] \neq 0_A$ for every $A \in R$.

Remark 3. Let r be a relation over a scheme context $\mathcal{C}_R$, and let g be a reality. As we previously discussed, $\mathcal{L}_r$ induces a closure operator from $\mathcal{L}_R$ to $\mathcal{L}_r$. Since $g(\mathcal{L}_r)$ is a closure system by Theorem 1, it also induces a closure operator from the powerset of R to $g(\mathcal{L}_r)$. In the remaining of this subsection, we write ϕ as the closure operator of $\mathcal{L}_r$ and ϕ_g as the closure operator of $g(\mathcal{L}_r)$ to avoid confusion.

Example 13. We illustrate in Figure 9 the six possible realities of our running example, along with the interpretations of the abstract lattice associated with the relation Patients (see Figure 6). Among these, g_2, g_3, g_5 and g_6 are strong realities. For example, we have $x_{g_1} = \langle e, e, c\rangle$, $x_{g_2} = \langle d_A, e, c\rangle$, and $x_{g_3} = \langle s, e, u\rangle$.

Now we give some properties of strong realities. First, unlike normal realities, there may be cases where $\mathcal{R}_s = \emptyset$, as shown by the following example.

Example 14. Let $R = \{A, B, C\}$ and $\mathcal{L}_A, \mathcal{L}_B, \mathcal{L}_C$ be the abstract lattices of Figure 10. Due to $\mathcal{L}_A$ no strong reality is possible: for any attribute interpretation h_A, each element a, b, c of $\mathcal{L}_A$ must be assigned a binary value so that at least two elements will have the same value, say a, b. However, the homomorphism property implies that $h_A(a \vee b) = h_A(a) \vee h_A(b) = h_A(a) \wedge h_A(b) = h_A(a \wedge b)$ so that either $h_A(0) = 1$ or $h_A(1) = 0$, a contradiction with the definition of an interpretation.

Let us assume that $\mathcal{R}_s$ is non-empty. We show that strong realities are related to prime/co-prime decomposition pairs in lattices [Mar92]. Let $\mathcal{L}$ be a lattice, and $x \in \mathcal{L}$. We say that x is *prime* if for any $y, z \in \mathcal{L}$, $y \wedge z \leq x$ implies either $y \leq x$ or $z \leq x$. The term *co-prime* is defined dually with $\vee$. The bottom element of $\mathcal{L}$ cannot be co-prime, and the top element cannot be prime. It is known from [Mar92, Theorem 6], that $\mathcal{L}$ has a prime

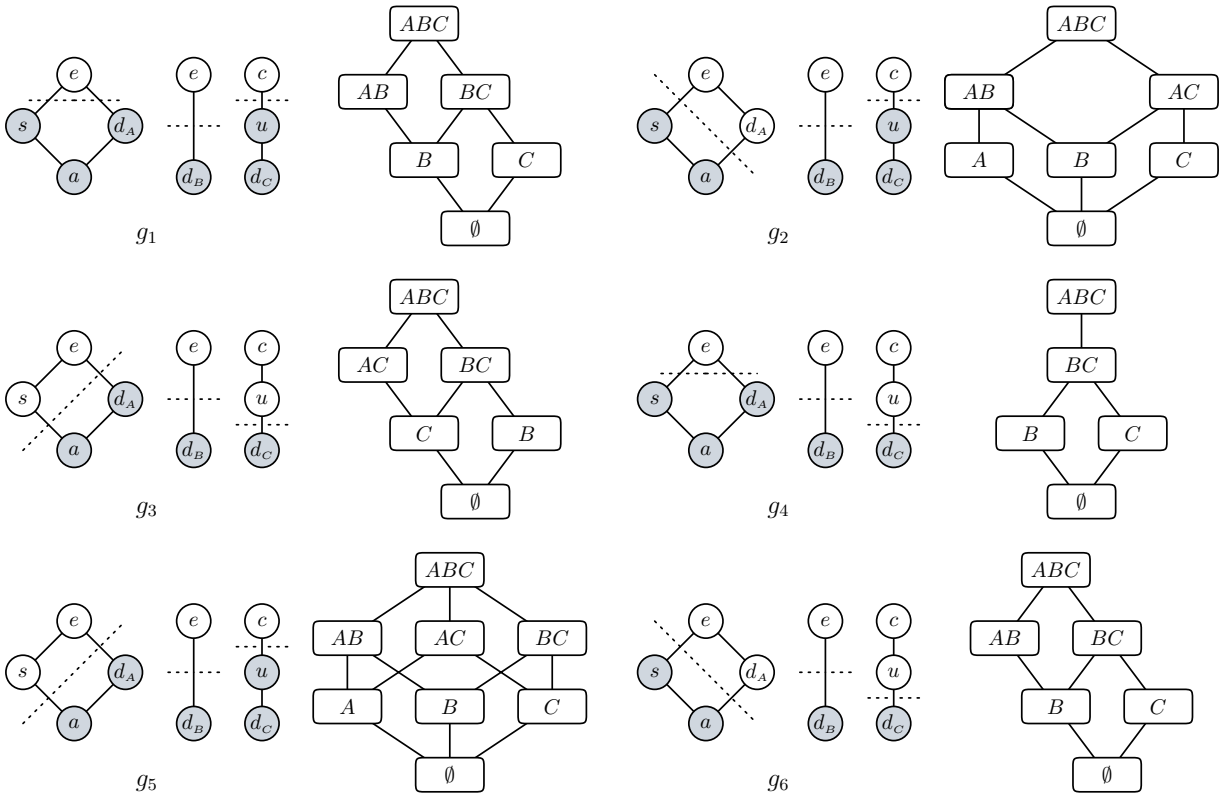


Figure 9: Realities and interpretations of the abstract lattice associated with the relation Patients

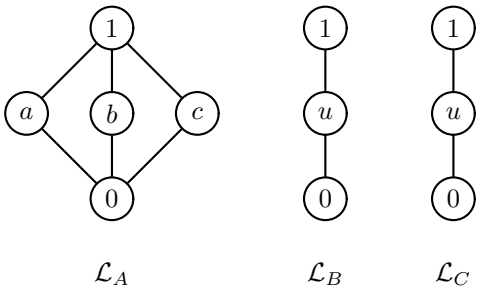


Figure 10: abstract lattices without strong realities

element p if and only if it has a co-prime element c. Actually, if p is prime, then it has a unique associated co-prime element $c = \mathsf{min}_{\leq}\{x \in \mathcal{L} \mid x \not\leq p\}$. Dually, if c is co-prime, it has a unique associated prime element $p = \mathsf{max}_{\leq}\{x \in \mathcal{L} \mid x \not\geq c\}$. This correspondence between primes and co-primes is bijective. Furthermore, a pair (c, p) splits $\mathcal{L}$ into two disjoint parts $\{x \in \mathcal{L} \mid x \geq c\}$ and $\{x \in \mathcal{L} \mid x \leq p\}$ such that $\{x \in \mathcal{L} \mid c \leq x\} \cup \{x \in \mathcal{L} \mid x \leq p\} = \mathcal{L}$. For any $A \in R$, if $\mathcal{L}_A$ has a pair c, p of coprime/prime elements, we shall denote it $(c, p)_A$. Furthermore, the set of co-prime (resp. prime) elements of $\mathcal{L}_A$ is denoted by $\mathsf{CPr}(\mathcal{L}_A)$ (resp. $\mathsf{Pr}(\mathcal{L}_A)$). Now, we establish the relationship between strong realities and co-prime/prime pairs.

Proposition 5. *Let $\mathcal{C}_R$ be a scheme context, and g a reality. Then, g is a strong reality if and only if for any $A \in R$, there are co-prime/prime pairs $(c, p)_A$ such that $c = \mathsf{min}_{\leq}\{x_A \in \mathcal{L}_A \mid g_{|A}(x_A) = 1\}$ and $p = \mathsf{max}_{\leq}\{x_A \in \mathcal{L}_A \mid g_{|A}(x_A) = 0\}$.*

Proof. We begin with the only if part. Let $g \in \mathcal{R}_s$ and consider $g_{|A}$ for some $A \in R$.

We show that $g_{|A}$ induces a co-prime/prime pair $(c,p)_A$. We put $c = \bigwedge g_{|A}^{-1}(1)$ and $p = \bigvee g_{|A}^{-1}(0)$. Note that as $g(0_A) = 0$ and $g(1_A) = 1$ by definition, $g_{|A}^{-1}(1)$ and $g_{|A}^{-1}(0)$ are non-empty. Furthermore, since g is increasing and homomorphic, both p and c are well-defined and satisfy $g_{|A}(p) = 0$ and $g_{|A}(c) = 1$. Let us prove that p is prime. Let $x_A, y_A \in \mathcal{L}_A$, such that $x_A \wedge y_A \leq p$. As $x_A \wedge y_A \leq p$, it must be that $g_{|A}(x_A \wedge y_A) = 0$ and either $g_{|A}(x_A) = 0$ or $g_{|A}(y_A) = 0$ since otherwise we would break the $\wedge$-homomorphic property of g. By construction of p then, it follows that either $x_A \leq p$ or $y_A \leq p$. A similar reasoning can be applied to c. We show that $\mathsf{min}_{\leq}\{x_A \in \mathcal{L}_A \mid x_A \not\leq p\}$ is a singleton element and that it equals c. Let $y_A, z_A \in \{x_A \in \mathcal{L}_A \mid x_A \not\leq p\}$. By construction of p, and since g is an homomorphism, it must be that $g_{|A}(y_A) = g_{|A}(z_A) = 1$ so that $y_A \wedge z_A \in \{x_A \in \mathcal{L}_A \mid x_A \not\leq p\}$. Therefore, this set has a unique minimum element. By definition of c, we also have that $c \leq y_A \wedge z_A$ and $c = \mathsf{min}_{\leq}\{x_A \in \mathcal{L}_A \mid x_A \not\leq p\}$ holds. Similarly we obtain $p = \mathsf{max}_{\leq}\{x_A \in \mathcal{L}_A \mid x \not\geq c\}$. Thus $(c,p)_A$ is indeed a co-prime/prime pair of $\mathcal{L}_A$. As these arguments can be applied for any $A \in R$, the only if part of the statement follows.

Now we move to the if part. Let $A \in R$ and $(c,p)_A$ be a co-prime/prime pair of $\mathcal{L}_A$. Let us put h_A as follows:

$$h_A(x_A) = \begin{cases} 1 & \text{if } x_A \geq c \\ 0 & \text{if } x_A \leq p \end{cases}$$

We have to check that such a definition covers the whole lattice $\mathcal{L}_A$ in a satisfying way. First, Let $x_A \in \mathcal{L}_A$ such that $x_A \not\leq p$. As p is prime and $(c,p)_A$ is a co-prime/prime pair, we have that $c = \mathsf{min}_{\leq}\{y_A \in \mathcal{L}_A \mid y_A \not\leq p\}$ so that $x_A \geq c$. Hence, any $x_A \in \mathcal{L}_A$ is uniquely mapped to 0 or 1 and h_A is an increasing mapping. As for the homomorphism property, we show the $\vee$-homomorphic side of h_A. As h_A is increasing, the case where $h_A(x_A) = h_A(y_A) = 1$ is already covered. However, if $h_A(x_A) = h_A(y_A) = 0$, then we have $x_A \leq p$ and $y_A \leq p$ so that $x_A \vee y_A \leq p$ and hence $h_A(x_A \vee y_A) = 0$ by definition of h_A. The $\wedge$-homomorphic side of h_A is proved analogously. Now, if we consider $g = \langle h_{A_1}, \ldots, h_{A_n} \rangle$ we obtain a strong reality in virtue of Proposition 1 and Definition 6. □

Thus, any strong reality is the result of the choice of a co-prime/prime pair in each abstract lattice. More formally, for any $A \in R$ and any $c \in \mathsf{CPr}(\mathcal{L}_A)$, let us define $h_A^c(x_A) = 1$ if $x_A \geq c$ and 0 if $x_A \leq p$ where p is the prime element associated with c in $\mathcal{L}_A$. Then, we have:

$$\mathcal{R}_s = \prod_{A \in R} \{h_A^c \mid c \in \mathsf{CPr}(\mathcal{L}_A)\}$$

With (strong) realities, we can now study the relationship between abstract FDs and classical FDs. As we have seen, any relation is associated with an abstract lattice from which abstract FDs can be derived. Furthermore, any (strong) reality gives a semantic for classical FDs when the abstract lattice is interpreted. First, we settle the notion of satisfaction of an abstract FD through a reality.

Definition 7 (Satisfaction)**.** Let $\mathcal{C}_R$ be a scheme context, r a relation over $\mathcal{C}_R$, g a reality and $x \to y$ an abstract FD. The functional dependency $g(x) \to g(y)$ is satisfied in r, denoted by $r \models_{\mathcal{C}_R,g} g(x) \to g(y)$ (or simply $r \models_g g(x) \to g(y)$) if for all $t_1, t_2 \in r$, $g(x) \subseteq g(f_R(t_1,t_2))$ implies $g(y) \subseteq g(f_R(t_1,t_2))$.

Note that in the interpretation $g(x) \to g(y)$ of an abstract FD $x \to y$, $g(y)$ may coincide with a set Y instead of an attribute A. Without loss of generality, a FD of the form $X \to Y$ should be read as the equivalent set of FDs $\{X \to A \mid A \in Y\}$.

We first give a proposition similar to Proposition 3. This allows us to work directly on the lattice $\mathcal{L}_r$ induced by r without loss of generality with respect to FDs.

Proposition 6. *Let r be a relation over $\mathcal{C}_R$, g a reality, and $x \to y$ an abstract FD. We have $r \models_g g(x) \to g(y)$ if and only if for any $z \in \mathcal{L}_r$, $g(x) \subseteq g(z)$ implies $g(y) \subseteq g(z)$, denoted $\mathcal{L}_r \models_g g(x) \to g(y)$.*

In what follows, we will need the next definition.

Definition 8. Let g be a reality. The *projection* $\pi_g(x)$ of $x \in \mathcal{L}_R$ on g is given as follows: for any $A \in R$, $\pi_g(x)[A] = x_g[A]$ if $x[A] \geq x_g[A]$, and 0_A otherwise.

We prove in Theorem 2 that the interpretation of a valid abstract FD of $\mathcal{L}_r$ is true for g if its projection on g is still a valid abstract FD for $\mathcal{L}_r$. We need the following technical lemma first.

Lemma 1. *Let r be a relation over a scheme context $\mathcal{C}_R$. Let $x \in \mathcal{L}_R$, and g be a reality. Then $g(\phi(\pi_g(x))) = \phi_g(g(\pi_g(x))) = \phi_g(g(x))$.*

Theorem 2. *Let r be a relation over a scheme context $\mathcal{C}_R$. Let g be a reality, and $x \to y$ an abstract FD. Then $\mathcal{L}_r \models_g g(x) \to g(y)$ if and only if $\mathcal{L}_r \models \pi_g(x) \to \pi_g(y)$.*

Proof. Let g be a reality, and let $x \to y$ be an abstract FD. We prove the only if part. Let us assume that $\mathcal{L}_r \models_g g(x) \to g(y)$. As $g(x) = g(\pi_g(x))$ and $g(y) = g(\pi_g(y))$, we also have that $\mathcal{L}_r \models_g g(\pi_g(x)) \to g(\pi_g(y))$. We show that $\mathcal{L}_r \models \pi_g(x) \to \pi_g(y)$. Since $\mathcal{L}_r \models_g g(\pi_g(x)) \to g(\pi_g(y))$, we have that $g(\pi_g(y)) \subseteq \phi_g(g(\pi_g(x))) = g(\phi(\pi_g(x)))$ by Lemma 1. We prove that $\pi_g(y) \leq \phi(\pi_g(x))$. Let $A \in R$. If $A \notin g(\pi_g(y))$, then we have $\pi_g(y)[A] = 0_A$ by construction of π_g and hence $\pi_g(y)[A] \leq_A \phi(\pi_g(x))[A]$. If $A \in g(\pi_g(y)) \subseteq g(\phi(\pi_g(x)))$, then $\pi_g(y)[A] \leq_A \phi(\pi_g(x))[A]$ holds again by construction of π_g. Therefore, $\pi_g(y) \leq \phi(\pi_g(x))$ and $\mathcal{L}_r \models \pi_g(x) \to \pi_g(y)$ follows.

We move to the if part. Let us assume that $\mathcal{L}_r \models \pi_g(x) \to \pi_g(y)$. We have that $\pi_g(y) \leq \phi(\pi_g(x))$. Then, $g(\pi_g(y)) \subseteq g(\phi(\pi_g(x)))$ as g is increasing, and $g(\phi(\pi_g(x))) = \phi_g(g(\pi_g(x))) = \phi_g(g(x))$ by Lemma 1. As $g(\pi_g(y)) = g(y)$, we have that $g(y) \subseteq \phi_g(g(x))$ and $\mathcal{L}_r \models_g g(x) \to g(y)$ follows. □

Moreover, if an abstract FD holds, then it reflects a potential dependency on r, meaning that there should be a reality which turns it into a valid FD. This is the aim of the next proposition.

Proposition 7. *Let $\mathcal{C}_R$ be a schema context, r be a relation over $\mathcal{C}_R$, and $x \to y$ be an abstract FD. If $r \models x \to y$, then there exists a reality g such that $r \models_g g(x) \to g(y)$.*

Proof. Assume that $r \models x \to y$. By Proposition 6, this is equivalent to $\mathcal{L}_r \models x \to y$ where $\mathcal{L}_r$ is the abstract lattice associated with r. Let g be a reality satisfying $\pi_g(x) = x$. Note that such a reality always exists as it only requires to put $x_g[A] = x[A]$ for every $A \in R$ such that $x[A] \neq 0_A$. Then, by Lemma 1, $g(\phi(x)) = \phi_g(g(x)$. Since $\mathcal{L}_r \models x \to y$, we have that $y \leq \phi(x)$ and therefore $g(y) \subseteq g(\phi(x)) = \phi_g(g(x))$ so that $\mathcal{L}_r \models_g g(x) \to g(y)$. We deduce that $r \models g(x) \to g(y)$ as expected. □

To conclude this subsection, we illustrate Proposition 7 on our running example.

Example 15. We consider our running example. Let $x \to y$ be the abstract FD where $x = \langle e, d_B, u \rangle$ and $y = \langle s, e, d_C \rangle$. It is a valid abstract FD of the lattice associated to Patients (and hence of Patients). Thus, according to Proposition 7, there must exist a reality g for which the FD $g(x) \to g(y)$ is valid in r. This is the case, for instance, of the reality g_1 depicted to the left of Figure 2 (see also Figure 9). More precisely, we have $g_1(x) = A$, $g_1(y) = B$, and $r \models_{g_1} A \to B$.

5.2 Possible and certain FDs

In this part, we are interested in functional dependencies from another point of view. We are given a relation r over a scheme context $\mathcal{C}_R$, and a functional dependency $X \to A$. Let us remind that, in our terms, a functional dependency $X \to A$ means a pair (X, A) where $X \subseteq R$ and $A \in R$. Since any reality g maps $\mathcal{L}_r$ to a closure system, it is natural to wonder about the functional dependencies holding in $g(\mathcal{L}_r)$. Beforehand, we formally define the meaning of a FD holding through a reality. This will mainly be a reformulation of Definition 7 with attribute sets instead of abstract tuples.

Definition 9. Let r be a relation over $\mathcal{C}_R$, g a reality, $X \subseteq R$, and $A \in R$. Then, $X \to A$ is satisfied in r with respect to $\mathcal{C}_R$ and g, denoted $r \models_g X \to A$, if for every $t_1, t_2 \in r$, $X \subseteq g(f_R(t_1, t_2))$ implies $A \in g(f_R(t_1, t_2))$.

Proposition 3 can also be reformulated in this way, thus introducing the notation $\mathcal{L}_r \models_g X \to A$ if for any $x \in \mathcal{L}_r$, $X \subseteq g(x)$ implies $A \in g(x)$.

Proposition 8. *Let r be a relation over $\mathcal{C}_R$, g a reality, $X \subseteq R$, and $A \in R$. We have $r \models_g X \to A$ if and only if $\mathcal{L}_r \models_g X \to A$.*

Proof. Let $x, a \in \mathcal{L}_R$ such that $g(x) = X$ and $g(a) = A$. Note that x, a must exist as $g(\mathcal{L}_R) = 2^R$. The result follows from 3 applied to x, a. □

There are many different realities for a given $\mathcal{C}_R$. Therefore, the same FD $X \to A$ may or may not hold depending on the reality we choose. However, there are also some FDs that will hold for every reality such as the trivial case $X \to A$, $A \in X$. Thus, inspired by certain query answering, we define certain functional dependencies as those that are true for any reality. Similarly, a functional dependency is possible if it holds for at least one reality. They are defined along with decision problems.

Definition 10 (certain FD, strongly certain FD)**.** Let r be a relation over $\mathcal{C}_R$, and $X \to A$ a functional dependency. We say that $X \to A$ is *certain* (resp. *strongly certain*) if for any reality $g \in \mathcal{R}$ (resp. strong reality $g \in \mathcal{R}_s$), $r \models_g X \to A$ holds.

Certain Functional Dependency (CFD)

Input:	A scheme context $\mathcal{C}_R$, a relation r over $\mathcal{C}_R$, and a functional dependency $X \to A$.
Question:	Is $X \to A$ certain with respect to $r, \mathcal{C}_R$?

Strongly Certain Functional Dependency (SCFD)

Input:	A scheme context $\mathcal{C}_R$, a relation r over $\mathcal{C}_R$, and a functional dependency $X \to A$.
Question:	Is $X \to A$ strongly certain with respect to $r, \mathcal{C}_R$?

Definition 11 (possible FD, strongly possible FD)**.** Let r be a relation over $\mathcal{C}_R$, and $X \to A$ a functional dependency. We say that $X \to A$ is *possible* (resp. *strongly possible*) if there exists a reality $g \in \mathcal{R}$ (resp. a strong reality $g \in \mathcal{R}_s$) such that $r \models_g X \to A$ holds.

Possible Functional Dependency (PFD)

Input:	A scheme context $\mathcal{C}_R$, a relation r over $\mathcal{C}_R$, and a functional dependency $X \to A$.
Question:	Is $X \to A$ possible with respect to $r, \mathcal{C}_R$?

Strongly Possible Functional Dependency (SPFD)

Input:	A scheme context $\mathcal{C}_R$, a relation r over $\mathcal{C}_R$, and a functional dependency $X \to A$.
Question:	Is $X \to A$ strongly possible with respect to $r, \mathcal{C}_R$?

Remark 4. All the complexity statements of this section are based on the complexity assumption given after Definition 1.

Before discussing the complexity of these problems we give an illustration of possible and certain functional dependencies.

Example 16. We continue the running example. In Table 5, we list the (left-minimal) non-trivial FDs holding in Patients, according to the realities g_1 to g_6 given in Figure 9. It follows that all of them are possible, since they hold in at least one reality. On the other hand, no non-trivial FD holds in every reality. Thus, certain FDs of Patients are of the form $X \to Y$ where $Y \subseteq X$.

realities	FDs holding in Patients
g_1	$A \to B$
g_2	$BC \to A$
g_3	$A \to C$
g_4	$A \to BC$
g_5	$\emptyset$
g_6	$A \to B$

Table 5: Non-trivial FDs holding in Patients, depending on the reality

Remark 5. A functional dependency $X \to A$ which is valid in the classical sense (with respect to equality) needs not be possible in a given scheme context. Whether or not $X \to A$ will be possible depends on how the comparability functions are given. In practice, this is not a problem: if the comparability functions provided by domain experts do not allow to identify classical functional dependencies, it means that usual FDs may not be relevant with respect to their data.

Now, we show that the number of realities can be exponential.

Proposition 9. *Let R be a relation scheme, and $\mathcal{C}_R$ a scheme context. Then, the number of (strong) realities can be exponential in the size of R and $\mathcal{C}_R$.*

Proof. Let R be a relation scheme and $\mathcal{C}_R = \{(A, f_A, \mathcal{L}_A) \mid A \in R\}$ where $\mathcal{L}_A$ is the three-element lattice $0 \leq u \leq 1$ for each $A \in R$. Note that the size of $\mathcal{C}_R$ is bounded by the size of R. Now, for each $\mathcal{L}_A$, there are only two possible interpretations h_A^1 and h_A^2:

1. $h_A^1(0) = h_A^1(u) = 0$ and $h_A^1(1) = 1$,
2. $h_A^2(0) = 0$ and $h_A^2(u) = h_A^2(1) = 1$.

Clearly both h_A^1 and h_A^2 are homomorphisms. Using Proposition 1 and Theorem 1 we can describe $\mathcal{R}$ as follows:

$$\mathcal{R} = \left\{ \langle h_{A_1}^{i_1}, h_{A_2}^{i_2}, \ldots, h_{A_n}^{i_n} \rangle \mid i_k \in \{1, 2\}, 1 \leq k \leq n \right\}$$

Thus, $\mathcal{R}$ is in bijection with all binary words of size n on the $\{1, 2\}$ alphabet. Hence $|\mathcal{R}| = 2^n$. □

It follows that the problems of deciding whether $X \to A$ is possible or certain cannot be solved in polynomial time with a brute force algorithm testing all the existing realities.

5.2.1 Certain functional dependencies

Now, we investigate the problems related to certain functional dependencies. Note that thanks to Proposition 8 we can use equivalently $r \models X \to A$ and $\mathcal{L}_r \models X \to A$. To do so, we characterize certainty of a FD using the structure of $\mathcal{L}_r$. As a reminder, $\mathsf{CPr}(\mathcal{L}_A)$ (resp. $\mathsf{Pr}(\mathcal{L}_A)$) denotes the set of co-prime (resp. prime) elements of the abstract lattice $\mathcal{L}_A$, for any $A \in R$.

Lemma 2. *Let r be a relation over $\mathcal{C}_R$, and $X \to A$ a functional dependency. Then:*

1. *$X \to A$ is certain if and only if for any $x \in \mathcal{L}_r$, either $x[A] = 1_A$ or there exists $B \in X$ such that $x[B] = 0_B$.*
2. *$X \to A$ is strongly certain if and only if for any $x \in \mathcal{L}_r$ either $x[A] \geq_A \bigvee \mathsf{CPr}(\mathcal{L}_A)$ or there exists $B \in X$ such that $x[B] \not\geq_B c$ for any $c \in \mathsf{CPr}(\mathcal{L}_B)$.*

Therefore, deciding whether a FD is certain amounts to checking the elements of $\mathcal{L}_r$. Let $X \subseteq R$. We define $\psi_X \in \mathcal{L}_R$ where $\psi_X[A] = 1_A$ if $A \in X$, and 0_A otherwise. In other words, ψ_X is the characteristic vector of X in $\mathcal{L}_R$. We deduce in the next theorem that CFD can be solved in polynomial time.

Theorem 3. CFD *can be solved in polynomial time.*

Proof. Let $(r, \mathcal{C}_R, X \to A)$ be an instance of CFD. From Lemma 2, we have to check that for any element x in $\mathcal{L}_r$, either $x[A] = 1_A$ or there exists $B \in X$ such that $x[B] = 0_B$. Let us set up $x \in \mathcal{L}_R$ as follows:

- $x[A] = 1_A$,
- $x[B] = 0_B$ for any $B \neq A$ (in particular for any $B \in X$)

We consider $\phi(x)$. Remark that as for any $B \in R$, $\mathcal{L}_B$ is given in the input, computing the $\wedge$ operation in $\mathcal{L}_R$ can be conducted in polynomial time in the size of $\mathcal{C}_R$ by running over the elements of $\mathcal{L}_B$.

Note that any element $y \in \mathcal{L}_r$ which satisfies $y[A] = 1_A$ does not require further verification and furthermore, lies above $\phi(x)$ by construction of x. Hence it remains to check that for any $y \not\geq \phi(x)$ in $\mathcal{L}_r$, there exists some $B \in X$ such that $x[B] = 0_B$. We show that we only need to test this property for the set $\mathsf{max}_{\leq}\{y \not\geq \phi(x) \mid y \in \mathcal{L}_r\}$. First, assume that there exists $m \in \mathsf{max}_{\leq}\{y \not\geq \phi(x) \mid y \in \mathcal{L}_r\}$ such that for any $B \in X$, $m[B] \neq 0_B$. Then, the condition of Lemma 2 fails with m and hence $X \to A$ is not certain. Now suppose that for any $m \in \mathsf{max}_{\leq}\{y \not\geq \phi(x) \mid y \in \mathcal{L}_r\}$, there exists some $B \in X$ such that $m[B] = 0_B$. By definition, for any $y \not\geq \phi(x)$ there must exists some $m \in \mathsf{max}_{\leq}\{y \not\geq \phi(x) \mid y \in \mathcal{L}_r\}$ such that $y \leq m$. As by assumption we supposed that there exists some $B \in X$ such that $m[B] = 0_B$ and as $y \leq m$, $y[B] = 0_B$ follows. Thus, for any $y \not\geq \phi(x)$ in $\mathcal{L}_r$, there exists some $B \in X$ such that $x[B] = 0_B$ if and only if for any $m \in \mathsf{max}_{\leq}\{y \not\geq \phi(x) \mid y \in \mathcal{L}_r\}$, there exists $B \in X$ such that $m[B] = 0_B$.

It remains to show that the set $\mathsf{max}_{\leq}\{y \not\geq \phi(x) \mid y \in \mathcal{L}_r\}$ can be checked efficiently. First, observe that $\mathsf{max}_{\leq}\{y \not\geq \phi(x) \mid y \in \mathcal{L}_r\} \subseteq \mathcal{M}(\mathcal{L}_r)$. Then, by construction of $\mathcal{L}_r$, any element y in $\mathcal{L}_r \setminus f_R(r)$ is obtained by taking the meet of a subset of $f_R(r)$. Thus,

$y \notin \mathcal{M}(\mathcal{L})$, and $\mathcal{M}(\mathcal{L}_r) \subseteq f_R(r)$. As the size of $f_R(r)$ is bounded by $|r^2|$ and each verification takes $O(|R|)$ operations, we conclude that checking all the tuples in $f_R(r)$ can be done in polynomial time in the size of $(r, \mathcal{C}_R, X \to A)$, thus concluding the proof. □

We prove a similar theorem for strongly certain functional dependencies. However, in this case, we have to add a pre-processing step to compute co-prime elements of the attribute lattice $\mathcal{L}_A$, for any $A \in R$.

Theorem 4. SCFD *can be solved in polynomial time.*

Proof. Let $(r, \mathcal{C}_R, X \to A)$ be an instance of SCFD. We first need to compute, for any $B \in R$, the set $\mathsf{CPr}(\mathcal{L}_B)$. As $\mathcal{L}_B$ is given in the input $\mathcal{C}_R$, this can be done in polynomial time in the size of $\mathcal{C}_R$ by greedily checking for each element x_B in $\mathcal{L}_B$ whether $\mathsf{max}_{\leq}\{y_B \in \mathcal{L}_B \mid y_B \not\geq x_B\}$ is a singleton set. Therefore, we can already check that strong realities exist by Proposition 5 in polynomial time. If there are no strong realities, the answer to SCFD is positive.

If there are strong realities, we use the arguments of Theorem 3 with $x \in \mathcal{L}_R$ as follows:

- $x[A] = \bigvee \mathsf{CPr}(\mathcal{L}_A)$,
- for any other $B \in R$, $B \neq A$, $x[B] = 0_B$.

Note that instead of checking whether for any $m \in \mathsf{max}_{\leq}\{y \not\geq \phi(x) \mid y \in \mathcal{L}_r\}$, there exists some $B \in X$ such that $m[B] = 0_B$, we check that there exists $B \in X$ such that $m[B] \not\geq c$ for any $c \in \mathsf{CPr}(\mathcal{L}_B)$. This amounts equivalently to assert that $m[B] < \bigwedge \mathsf{Pr}(\mathcal{L}_B)$ by definition of co-prime/prime pairs decomposition. Again, as $\mathcal{L}_B$ is part of the input, computing $\mathsf{Pr}(\mathcal{L}_B)$ and $\bigwedge \mathsf{Pr}(\mathcal{L}_B)$ can be done in polynomial time in the size of $(r, \mathcal{C}_R, X \to A)$. □

5.2.2 Possible functional dependencies

Now, we settle the complexity of the problems PFD and SPFD. Again, we need an intermediary structural lemma. It is the dual of Lemma 2.

Lemma 3. *Let r be a relation over $\mathcal{C}_R$, and $X \to A$ a functional dependency. Then:*

1. *$X \to A$ is not possible if and only if there exists $x \in \mathcal{L}_r$, such that $x[A] = 0_A$ and $x[B] = 1_B$ for any $B \in X$.*
2. *$X \to A$ is strongly possible if and only if there exists $c_X \in \mathcal{L}_R$ such that $c_X[B] = c$, for some c in $\mathsf{CPr}(\mathcal{L}_B)$, for every $B \in X$, $c_X[B] = 0_B$ for every $B \notin X$, and such that $\phi(c_X)[A] \geq c$ for some $c \in \mathsf{CPr}(\mathcal{L}_A)$.*

Hence, for a given FD $X \to A$, PFD can be solved by checking $\phi(\psi_X)$. As a reminder, ψ_X is the characteristic vector of X in $\mathcal{L}_R$, that is $\psi_X[A] = 1_A$ if $A \in X$, and 0_A otherwise.

Theorem 5. PFD *can be solved in polynomial time.*

Proof. Let $(r, \mathcal{C}_R, X \to A)$ be an instance of PFD. According to Lemma 3, $X \to A$ is not possible if and only if there is an element x in $\mathcal{L}_r$ such that $x[B] = 1_B$ for any $B \in X$ and $x[A] = 0_A$. Since $f_R(r)$ generates $\mathcal{L}_r$, we have that $\phi(\psi_X) = \bigwedge\{x \in f_R(r) \mid \psi_X \leq x\}$. Moreover, the abstract lattices of each attribute context are part of the input so that computing $f_R(r)$ and the $\wedge$ operation in each lattice can be conducted in polynomial time. Thus, we can compute the closure $\phi(\psi_X)$ of ψ_X (note that $\psi_X[A] = 0_A$) in polynomial time in the size of r and $\mathcal{C}_R$. Then, either $\phi(\psi_X)[A] = 0_A$ in which case $X \to A$ is not possible, or $\phi(\psi_X)[A] \neq 0_A$ and $X \to A$ is possible. □

However, for strong possibility, one might have to test an exponential number of closures since we have to check for any possible combination of co-prime elements. In fact, we prove in the following theorem that SPFD is intractable (reduction from 3SAT).

Theorem 6. SPFD *is NP-complete.*

Proof. First we show that SPFD belongs to NP. A certificate is a strong reality $g\colon \mathcal{L}_R \to \{0,1\}^n$ which can be represented by the abstract tuple x_g of polynomial size. Moreover, the satisfaction of a functional dependency $X \to A$ can be tested in polynomial time by computing $g(f_R(t,t'))$ for every pair of tuples t, t' in the input relation r. Thus, SPFD belongs to NP.

To show NP-hardness, we use a reduction from 3-SAT. Let $\varphi = \bigwedge_{j=1}^{m} C_j$, $m \in \mathbb{N}$, be a 3-CNF over a set of variables $V = \{x_1, \ldots, x_n\}$, $n \in \mathbb{N}$. We assume without loss of generality that no clause of φ contains both x_i and $\bar{x}_i$, for each $1 \leq i \leq n$. We construct a scheme context $\mathcal{C}_R$, a relation r and a functional dependency $X \to A$ such that φ is satisfiable if and only if there exists a strong reality g (in $\mathcal{C}_R$) such that $r \models_g X \to A$.

First, we introduce an attribute A_i for each $x_i \in V$. Then, we define a relation scheme $R = \{A_1, \ldots, A_{n+1}\}$ where A_{n+1} is yet another attribute. For each $A_i \in R$, we put $\mathsf{dom}(A_i) = \mathbb{N}$. Now we define a scheme context $\mathcal{C}_R$ as follows:

- for each $1 \leq i \leq n$, we put $\mathsf{dom}_{abs}(A_i) = \{0_i, a_i, \bar{a}_i, 1_i\}$, $\mathcal{L}_{A_i} = (\mathsf{dom}_{abs}(A_i)), \leq_i)$ where $\leq_i$ is defined by $0 \leq_i a_i \leq 1_i$ and $0 \leq_i \bar{a}_i \leq_i 1$. We define the comparability function f_i associated to A_i as follows:

$$f_i(x,y) = \begin{cases} 1_i & \text{if } x = y \\ a_i & \text{if } |x-y| = 1 \\ \bar{a}_i & \text{if } |x-y| = 2 \\ 0_i & \text{otherwise.} \end{cases}$$

  The attribute context of A_i is $(A_i, f_i, \mathcal{L}_{A_i})$.

- for A_{n+1}, we put $\mathsf{dom}_{abs}(A_{n+1}) = \{0_{n+1}, 1_{n+1}\}$, $\mathcal{L}_{A_i} = (\mathsf{dom}_{abs}(A_{n+1})), \leq_{n+1})$ where $\leq_{n+1}$ is defined by $0 \leq_{n+1} 1$. The comparability function f_{n+1} associated to A_{n+1} reads as follows:

$$f_{n+1}(x,y) = \begin{cases} 1_{n+1} & \text{if } |x-y| \neq 1 \\ 0_{n+1} & \text{if } |x-y| = 1 \end{cases}$$

  The attribute context of A_{n+1} is $(A_{n+1}, f_{n+1}, \mathcal{L}_{A_{n+1}})$.

Let $\mathcal{C}_R = \{(A_i, f_i, \mathcal{L}_{A_i}) \mid 1 \leq i \leq n+1\}$ be the resulting scheme context. Its comparability function is called f_R. Now, we construct a relation r over $\mathcal{C}_R$. To every clause C_j in φ, $1 \leq j \leq m$ we associate a subrelation r_j with two tuples t and t':

- for each $A_i, 1 \leq j \leq n$:
  - if $x_i \in C_j$, $t[A_i] = 3j-2$ and $t'[A_i] = 3j$,
  - if $\bar{x}_i \in C_j$, $t[A_i] = 3j-2$ and $t'[A_i] = 3j-1$,
  - $t[A_i] = t'[A_i] = 3j$ otherwise.
- $t[A_{n+1}] = 3j-1$ and $t'[A_{n+1}] = 3j$.

Let $r = \bigcup_{1 \leq j \leq m} r_j$. We consider the functional dependency $X \to A$ where $X = \{A_1, ..., A_n\}$ and $A = A_{n+1}$. Clearly, the size of the reduction is polynomial in the size φ. The strong realities of the abstract context coincide with the Cartesian product $\{a_i, \bar{a}_i\}^n$.

Let g be a strong reality and for every $1 \leq i \leq n+1$, let h_i be the projection of g on the attribute A_i, *i.e.* $h_i = g_{|A_i}$. First, we show that $r \models_g X \to A$ if and only if $r_j \models_g X \to A$ for each subrelation r_j, $1 \leq j \leq m$. The only if part is clear, since $r \models_g X \to A$ and $r_j \subseteq r$ entails $r_j \models_g X \to A$. For the if part, it is sufficient to show that any pair of tuples lying in different subrelations always agree on A with respect to g. Let t, t' be two tuples of r such that t, t' are in distinct subrelations r_j, r_k where $1 \leq j < k \leq m$ (resp.). By construction of r, the minimum value of $|t[A_{n+1}] - t'[A_{n+1}]|$ is reached when $t[A_{n+1}] = 3j$ and $t'[A_{n+1}] = 3k - 1$. As $k \geq j + 1$, we obtain $3k - 1 \geq 3j + 2$ and hence $|t[A_{n+1}] - t'[A_{n+1}]| \geq 2$. Therefore, we have $f_{n+1}(t[A_{n+1}], t'[A_{n+1}]) = 1_{n+1}$ by definition of f_{n+1} and $h_{n+1}(f_{n+1}(t[A_{n+1}], t'[A_{n+1}])) = 1$ for each (strong) reality g. Thus, for every reality, whenever two tuples t and t' disagree on the right-hand side of $X \to A$, they must belong to the same subrelation.

Now we prove that φ is satisfiable if and only if $X \to A$ is strongly possible. We begin with the only if part. Suppose that φ is satisfiable and let $\mu : V \to \{0, 1\}$ be a valid truth assignment of φ. We construct a strong reality g such that $r \models_g X \to A$. For every attribute context $(A_i, f_i, \mathcal{L}_{A_i})$, $1 \leq j \leq n$, we define the following interpretation h_i:

- $h_i(0_i) = 0$ and $h_i(1_i) = 1$,
- $h_i(a_i) = \mu(x_i)$ and $h_i(\bar{a}_i) = 1 - \mu(x_i)$.

For A_{n+1}, we define $h_{n+1}(0_{n+1}) = 0$ and $h_{n+1}(1_{n+1}) = 1$. Let $g : \mathcal{L}_R \to \{0, 1\}^n$ be the scheme interpretation $g : \mathcal{L}_R \to \{0, 1\}^n$ defined by $g(\langle x_1, \ldots, x_{n+1} \rangle) = \langle h_1(x_1), \ldots, h_{n+1}(x_{n+1}) \rangle$. For every $1 \leq j \leq n$, we have $h_i(a_i) \neq h_i(\bar{a}_i)$ which implies that h_i is an homomorphism. Therefore, g is a strong reality by Proposition 1 and Definition 6. We show that $r \models_g X \to A$. Using previous discussion, it is sufficient to prove that $r_j \models_g X \to A$ for all $1 \leq j \leq m$ to obtain $r \models_g X \to A$. Hence, let t, t' be the two tuples of r_j for some $C_j = (\ell_1 \vee \ell_2 \vee \ell_3)$ in φ. At least one literal in C_j, say ℓ_1, validates C_j with respect to μ. Since $\ell_1 \in \{x_i, \bar{x}_i\}$ for some $A_i \in R$, we have two cases:

1. $\ell_1 = x_i$. Then $t[A_i] = 3j - 2$, $t'[A_i] = 3j$ and $f_i(t[A_i], t'[A_i]) = \bar{a}_i$. Moreover, $h_i(\bar{a}_i) = 1 - \mu(x_i) = 0$ since $\mu(x_i) = 1$. Consequently, $h_i(f_i(t[A_i], t'[A_i])) = 0$ and $r_j \models_g X \to A$ holds as $A_i \in X$.

2. $\ell_1 = \bar{x}_i$. Then $t[A_i] = 3j - 2$, $t'[A_i] = 3j - 1$ and $f_i(t[A_i], t'[A_i]) = a_i$. Since $h_i(a_i) = \mu(x_i) = 0$, we deduce $h_i(f_{A_i}(t[A_i], t'[A_i])) = 0$ and $r_j \models_g X \to A$ follows.

In other words, the tuples of every subrelation r_j cannot agree on X. We conclude that $X \to A$ is always satisfied in the strong reality g and that $X \to A$ is strongly possible.

We move to the if part. Assume that $X \to A$ is strongly possible and let g be a strong reality satisfying $r \models_g X \to A$. Since g is a strong reality and by Proposition 1, we have $h_i(a_i) \neq h_i(\bar{a}_i)$. As a consequence, we can define a truth assignment $\mu \colon V \to \{0, 1\}$ as follows:

$$\mu(x_i) = \begin{cases} 1 & \text{if } f_i(t[A_i], t[A'_i]) = \bar{a}_i \text{ for some } t, t' \in r \text{ and } h_i(\bar{a}_i) = 0 \\ 0 & \text{if } f_i(t[A_i], t[A'_i]) = a_i \text{ for some } t, t' \in r \text{ and } h_i(a_i) = 0 \\ 0 & \text{otherwise.} \end{cases}$$

Note that $f_i(t[A_i], t[A_i']) \in \{a_i, \bar{a}_i\}$ if and only if t, t' are the two tuples of the same sub-relation r_j, for some $1 \leq j \leq m$. We show that μ is a satisfying assignment of φ. Let C_j be a clause of φ and let t, t' be the two tuples of r_j. By assumption, $r_j \models_g X \to A$. Moreover, $h_{n+1}(f_{n+1}(t[A_{n+1}], t'[A_{n+1}])) = 0$ by construction of f_{n+1}. Therefore, there must exist $A_i \in X$ such that $h_i(f_i(t[A_i], t'[A_i])) = 0$. We have two cases:

- $f_i(t[A_i], t'[A_i]) = a_i$ in which case $\bar{x}_i$ belongs to C_j by construction of f_R. As $h_i(a_i) = 0$, we have $\mu(x_i) = 0$ by definition. Hence C_j is satisfied by μ.
- $f_i(t[A_i], t'[A_i]) = \bar{a}_i$ so that x_i belongs to C_j. Since $h_i(\bar{a}_i) = 0$, $\mu(x_i) = 1$ and C_j is satisfied by μ.

In any case, μ is a satisfying truth assignment for C_j. Applying the same reasoning to every clause of φ, we deduce that μ is a satisfying truth assignment for φ, concluding the proof. □

We summarize our complexity results in Table 6.

	Realities	Strong Realities
Certain FD	P (Theorem 3)	P (Theorem 4)
Possible FD	P (Theorem 5)	NP-complete (Theorem 6)

Table 6: Complexity results for Possible and Certain FD problems with respect to realities and strong realities

To conclude this section, we briefly discuss the complexity of the possibility and certainty problems for sets of FDs. Let $\mathcal{C}_R$ be a scheme context, r be a relation over $\mathcal{C}_R$ and $\mathbf{F}$ be a set of FDs also over $\mathcal{C}_R$. For a given reality g, we write $r \models_g \mathbf{F}$ if $r \models_g X \to A$ for each $X \to A$ in $\mathbf{F}$. If there exists a (strong) reality g such that $r \models_g \mathbf{F}$, then $\mathbf{F}$ is (strongly) possible in r. If $r \models_g \mathbf{F}$ for any (strong) reality g, then $\mathbf{F}$ is (strongly) certain in r.

For certainty, it is clear that $\mathbf{F}$ is (strongly) certain if and only if each $X \to A$ is (strongly) certain itself. Therefore, using Theorem 3 and Theorem 4, we directly deduce that it takes polynomial time to check that $\mathbf{F}$ is (strongly) certain.

We move to possibility. Deciding whether a single FD $X \to A$ is strongly possible is already NP-complete due to Theorem 6. It readily follows that checking whether $\mathbf{F}$ is strongly possible is also NP-complete. The case of simple possibility however seems harder to settle. To date, it remains an intriguing open question. we give some hints on its hardness. Formally, the problem reads as follows:

Possible Set of Functional Dependencies (PSFD)

Input:	A scheme context $\mathcal{C}_R$, a relation r and a set $\mathbf{F}$ of functional dependencies, both over $\mathcal{C}_R$.
Question:	`yes` if there exists a reality g such that $r \models_g \mathbf{F}$, `no` otherwise.

In order to study the complexity of PSFD, one could try to characterize the case where a set of FDs is not possible, much as in Lemma 3 for a single FD. For a single FD $X \to A$, the fact that $X \to A$ must be false in each reality allows us to identify an abstract tuple which characterizes the possibility of the FD $X \to A$. Unfortunately, this approach does not seem to apply to a set $\mathbf{F}$ of FDs, since there is in general no FD $X \to A$ of $\mathbf{F}$ which is

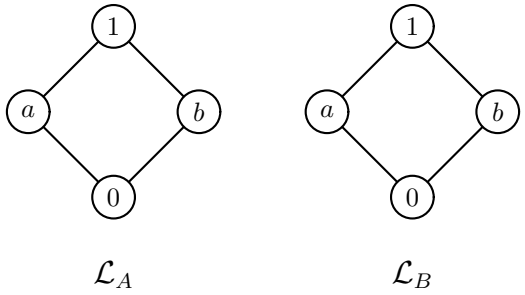
false in each reality (see Example 17). This makes a characterization similar to Lemma 3 harder to obtain.

Another strategy is based on the observation that $\mathbf{F}$ cannot be possible if there is some $X \to A$ in $\mathbf{F}$ which is not possible. However, unlike for certain FDs, the other direction is not true. More precisely, it may happen that all FDs in $\mathbf{F}$ are possible as singletons, but that $\mathbf{F}$ as a whole is not possible. Again, Example 17 illustrates this situation. As a consequence, using a decomposition of $\mathbf{F}$ and solve sub-problems to obtain an answer to PSFD on $\mathbf{F}$ seems unpromising. This observation also suggests that in general, if one obtains a valid reality for each FD of $\mathbf{F}$ (or for each part of some partition of $\mathbf{F}$), there is no guarantee that these realities can be combined to obtain a valid reality for $\mathbf{F}$. This is illustrated in Example 17.

Example 17. Let $\mathcal{C}_R = \{(A, f_A, \mathcal{L}_A), (B, f_B, \mathcal{L}_B)\}$ be a scheme context where $\mathsf{dom}(A) = \mathsf{dom}(B) = \mathbb{N} \cup \{\texttt{null}\}$, $\mathcal{L}_A$, $\mathcal{L}_B$ are the abstract lattices given in Figure 11, and f_A, f_B are comparability functions defined as follows:

$$f_A(u,v) = f_B(u,v) = \begin{cases} 1 & \text{if } u = v \neq \texttt{null} \\ a & \text{if } u \neq \texttt{null}, v \neq \texttt{null} \text{ and } u \neq v \\ b & \text{if } u = v = \texttt{null} \\ 0 & \text{otherwise.} \end{cases}$$

Let r be the relation on $\mathcal{C}_R$ given to the left of Figure 12. Its abstract lattice $\mathcal{L}_r$ is

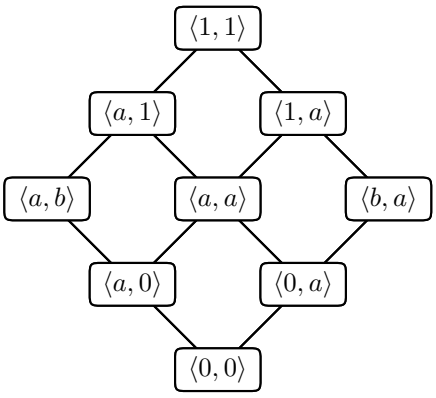


$\mathcal{L}_A$ $\mathcal{L}_B$

Figure 11: The abstract lattices of $\mathcal{C}_R$

represented to the right. We consider the set of FDs $\mathbf{F} = \{A \to B, B \to A\}$. For convenience

r	A	B
t_1	1	null
t_2	2	null
t_3	null	1
t_4	null	2



$\mathcal{L}_r$

Figure 12: A relation r and its associated abstract lattice $\mathcal{L}_r$

we give in Table 7 the list of all possible realities along with the abstract tuples which will be interpreted as counter-examples to $A \to B$ or $B \to A$. Explicitly, a set in $g(\mathcal{L}_r)$ is a counter-example to a FD $X \to A$ if it contains X but not A.

First, remark that both $A \to B$ and $B \to A$ are possible. Indeed, if we set $g = \langle b, a\rangle$ or $g = \langle a, 1\rangle$, then $r \models_g A \to B$ as there are no counter-examples in the resulting closure system. Similarly $r \models_g B \to A$ if $g = \langle a, b\rangle$ or $\langle a, 1\rangle$. However, there is no reality in which both FDs hold true together. Therefore, **F** is not possible in r. We can also make the following observations:

- among all the counter-examples of $A \to B$, none of them appears as a counter-example in all realities where $A \to B$ does not hold. As a consequence, it is not sufficient to exhibit a unique prototypical counter-example to each FD (as in Lemma 3) to characterize the possibility of **F**.
- the realities $\langle a, 1\rangle$ and $\langle 1, a\rangle$ make $B \to A$ and $A \to B$ valid (respectively). Moreover both $\langle a, 1\rangle \wedge \langle 1, a\rangle$ and $\langle a, 1\rangle \vee \langle 1, a\rangle$ are realities. But since **F** is not possible, neither of them are valid realities for **F**. This suggests that the rather natural process of combining realities with lattice operations is not sufficient to determine the possibility of **F**.

realities	counter-examples to $A \to B$	counter-examples to $B \to A$
$\langle a, a\rangle$	$\langle a, b\rangle, \langle a, 0\rangle$	$\langle b, a\rangle, \langle 0, a\rangle$
$\langle a, b\rangle$	$\langle 1, a\rangle, \langle a, a\rangle, \langle a, 0\rangle$	$\emptyset$
$\langle a, 1\rangle$	$\langle a, b\rangle, \langle a, a\rangle, \langle 1, a\rangle, \langle a, 0\rangle$	$\emptyset$
$\langle b, a\rangle$	$\emptyset$	$\langle a, a\rangle, \langle a, 1\rangle, \langle 0, a\rangle$
$\langle b, b\rangle$	$\langle 1, a\rangle, \langle b, a\rangle$	$\langle a, 1\rangle, \langle a, b\rangle$
$\langle b, 1\rangle$	$\langle 1, a\rangle, \langle b, a\rangle$	$\langle a, 1\rangle, \langle a, b\rangle$
$\langle 1, a\rangle$	$\emptyset$	$\langle b, a\rangle, \langle a, a\rangle, \langle a, 1\rangle, \langle 0, a\rangle$
$\langle 1, b\rangle$	$\langle 1, a\rangle$	$\langle a, 1\rangle$
$\langle 1, 1\rangle$	$\langle 1, a\rangle$	$\langle a, 1\rangle$

Table 7: Table of counter-examples to **F** according to the different realities

6 Related work

Incomplete information has been extensively studied in the database and artificial intelligence communities, see for example [AGNP18, AHV95, Ber11, CGL19, GPW14, ILJ89, Len02, LP19, SORK11]. Numerous research on data dependencies has been conducted over the last years, leading to a plethora of propositions from seminal FDs to more elaborated forms of dependencies, among which we quote [BCKN18, BKL13, CDP15, DLM92, Gog67, LP19, Ng01].

Many papers have studied the lattice representation of functional dependencies, such as [Day92, DLM92], which has been extended to multivalued dependencies by Balcazar and Baixeries in [BiJBN05]. However, they do not consider incomplete information as we do in this paper. W. Ng [Ng01] defined ordered domains in the relational data model, i.e., a partial order over every attribute domain is permitted. The paper studies the

consequences on both FDs and relational languages (algebra and SQL). His work does not consider incomplete information as we do: our partial order is not defined on the attribute domain, but on the abstract domain of attributes, and is required to form a lattice. It offers a new point of view on FDs in presence of incomplete information.

In [BCKN18], order dependencies are based on a transitive relation, and approximate dependencies on a symmetric relation, leading to approximate-matching dependencies. Unlike our framework, the comparison of two values in their model is a one-step process based on Boolean similarity functions (reflexive and symmetric). The set with similarities of Bauer and Hajdinjak [HB09] are close to attribute contexts, but the authors do not consider interpretations, neither do they study functional dependencies in this setup. In [BKL13], the authors study matching dependencies using Boolean similarity functions and matching functions (idempotent, commutative, and associative). More recently, the authors of [SPKN20] study matching dependencies where the similarity functions have values in the interval $[0, 1]$. The aim of matching functions is to chase the relation instance to obtain a clean relation. However, our way of dealing with incomplete information is completely different. In [Lib16], the author studies the semantics of SQL query answering in presence of incomplete information. He defines a multi-valued logic similar to our contribution, without considering data dependencies. Based on SQL three-valued model, a semantic for possible/certain (or weak/strong) FDs have been studied in several works [AAS22, KLLZ16, KL18, LL98, Lie82]. These works either rely on a completion of the data, or implicit interpretations (in our sense) of the similarity of two values at least one of which is `null` (see also the very recent work [LP20]). In our contribution, we do not modify the input data, and the interpretation of the comparison with a `null` value as true or false (0 or 1) in the SQL model appears as a particular case of our construction (see Section 4).

To evaluate the truth of a proposition like equality of two entities, it is usual to order the truth values in a lattice [AA96]. In addition to the well-known true/false lattice, several other semantics that might be useful have been studied. This is the case, for example, of Kleene or Łukasiewicz 3-valued logic [BB13], and Belnap's 4-valued logic, where truth values can be ordered by both their degree of truth and their degree of knowledge, or informativeness [Bel77]. Belnap's logic has been generalized to bi-lattices that are more suitable for non-monotonic reasoning [AA96, Gin88]. In [CGL16], the authors define many-valued semantics and informativeness ordering for query answers in incomplete databases with missing values represented by marked `null`. Recently, Grahne [GM18] used the 4-valued logic to capture inconsistencies in databases and for query answering in the presence of inconsistencies.

Several works extend the Codd's relational model using fuzzy logic. Among these, we quote [BP82, PT84, RM88, CKV92, BN93, CV94, TSDT05, Kis90, YOJ99, BPU99, BV11, BV18]. We redirect the reader to the survey of Jevskova et al. [JCE17] for an in-depth comparison of all these approaches. They replace the classical Boolean logic at the core of the logical foundations of the relational model by a fuzzy setup. More precisely, they replace the true/false Boolean lattice by a more complex set of truth values, still ordered in a lattice. Usually, this lattice is a (continuous) chain lattice on the interval $[0, 1]$, except for the framework of Belohlavek and Vychodil which consider every lattice that can be residuated, being more general [BV11, BV18]. Then, they adapt the logical operations of conjunction and implication to this new set of truth values, thus obtaining a fuzzy expression of the modus ponens (see [Bel12] for detailed explanations). Classic examples of such operations are Gödel, Goguen or Łukasiewicz operations. The lattice of truth values combined with these new operations form a residuated lattice. All these works

then study the extension of the classical functional dependencies within their fuzzy setup, the so-called fuzzy or generalized functional dependencies.

Our approach in this paper is different though: it comes on top of the relational model and we do not alter its logical foundations. Thus, it remains crisp. Indeed, the abstract lattices we define for each attribute are not the support for new logical operations, neither do they replace the underlying Boolean truth values. Abstract values can be seen as similarity values rather than truth values.

As a consequence, we can identify some typical examples where our framework will differ from the fuzzy setup:

- here, the result of a logical operation remains true or false. For instance, if we have two tuples t_1, t_2 at hand, the result of $f_A(t_1[A], t_2[A]) = x$ AND $f_B(t_1[B], t_2[B]) = y$ is either `true` or `false` (which has in general nothing to do with values in $\mathcal{L}_A$, $\mathcal{L}_B$). In the fuzzy set up, the result of such operation would be a value x in the underlying residuated lattice also containing y and z.

- abstract FDs are different in nature from fuzzy FDs and logical expressions $x \to y$ in the fuzzy set up. An abstract FD is a lattice expression [Day92] over the product of the abstract lattices used to replace equality. It is either true or false, depending on the closure operator induced by the data at hand. Thus, an abstract FD differs from a logical expression $x \to y$ (possibly with hedge) in a residuated lattice whose truth value is another value z in the residuated lattice, independent from the data [Bel12]. It also differs from similarity based fuzzy functional dependencies (SBFDs) since these latter are expressed in terms of (crisp in our case) attribute sets [BV18].

- Classic functional dependencies. In our work, the validity of a functional dependency in a relation depends on the choice of an interpretation. Interpretations are a generalization of α-cuts of fuzzy logic (in the interval $[0, 1]$) to any lattice. In the fuzzy relational model, FDs are restriction of SBFDs to nonranked data tables equipped with equality.

All the fuzzy models we mentioned do not consider comparability as a two-step process as we do in this paper. As a consequence, they do not consider the question of different interpretations, in particular the problem of characterizing (strong) realities. Similarly, they do not investigate (strongly) possible/certain functional dependencies.

At last, the fact that we rely on the classical logic leads to different implementation perspectives. Our framework can be implemented as a plugin on top of a RDBMS by extending DDL and SQL syntax with ways to declare comparabilities and lattices. On the other hand, implementing the fuzzy relational model amounts instead to propose a new DBMS, possibly based on existing RDBMS [BOV07].

One can note that the fuzzy set-up has also been applied to Formal Concept Analysis (FCA) [Bel12, BV05, BJFG94, BFG00, MOARC09, MOA10]. The usual incidence relation is replaced by a (possibly residuated) lattice of truth values equipped with specific logical connectives. These connectives are used to derive fuzzy concepts and fuzzy attribute implications, in the spirit of fuzzy functional dependencies for fuzzy relational systems [BV18]. However, since the principles are similar to fuzzy relational systems for databases, and our framework is database-oriented, we do not develop fuzzy FCA further.

Many contributions have been proposed to deal with uncertain FDs, among which we quote [BKL13, LP19, SC11], and [CDP15] for a survey. The authors of [KLTV06] have considered minimal relaxations for query conditions involving comparison operators (as

selection or join queries) to obtain an answer. Their works concern numerical attributes where relaxation is quantified as value differences, but this can also be applied to any *"blackbox"* function that quantifies the relaxation. Our contribution is an extension of these works within a declarative perspective, using lattice theory.

7 Conclusion

Defining equality between two tuples is a central problem in databases, notably due to data quality problems such as missing values and uncertainty. In practice, domain experts should be able to easily give a meaningful semantics for equality. In this paper, we have introduced a lattice-based formalism allowing to deal with many possible interpretations of data value equalities. Our approach is able to handle both missing and uncertain values, which might be of practical use in applications where values are known to be imprecise and dirty. In order to determine whether two tuples are equal, they are first compared using a comparability function which returns an abstract tuple representing their similarity. Then, an interpretation maps this abstract tuple to a binary vector where 1 means equality and 0 difference. We introduced realities as particular interpretations satisfying two reasonable consistency rules: they are increasing and guarantee that meet of two abstract values considered equal (interpreted as 1) is also interpreted as 1. Strong realities further satisfy the rule that the upper bound of two abstract values considered different (interpreted as 0) is also interpreted as 0.

We studied this framework with respect to functional dependencies. In this setup, we associate an abstract lattice with any relation. This abstract lattice naturally induces abstract functional dependencies. We showed that realities turn the abstract lattice of a relation into a closure system over the relation scheme. Furthermore, we exhibited a relationship between abstract FDs and FDs based on realities. On the other hand, we studied the problem of deciding whether a functional dependency over an incomplete database is certain (it holds in all possible worlds represented by the realities) or possible (there exists a possible world in which it holds).

Future work include both theoretical and practical contributions. On the practical point of view, our framework could be instantiated on top of DBMS in order to provide a declarative view of equality. On this purpose, it would be necessary to slightly extend SQL and DDL syntax to allow the user to declare comparability functions, abstract lattices and realities only on the relevant attributes. In this perspective, a query relying on realities would be rewritten in order to use comparability functions on attributes where they are defined (otherwise, equality is kept). In case a query does not use any keyword, it would be interpreted as a classical query. There is an ongoing work on SQL queries based on these principles within the context of a collaboration with the (CEMAFROID).

On the theoretical side we plan to investigate in greater depths abstract dependencies and their interactions with realities. For example, if we are given a cover of an abstract lattice by abstract FDs, is it true that for any reality, the interpretation of this cover will contain a cover of the lattice interpretation? Fascinating questions are also pending regarding possible functional dependencies. For instance, can we decide in polynomial time that there exists a reality in which a given set of FDs holds? There are also pending questions regarding functional dependencies. In particular, the complexity of deciding whether a set of functional dependencies is possible remains an intriguing open question.

Acknowledgments

The first author is funded by the French government IDEXISITE initiative 16-IDEX-0001 (CAP 20-25). The third author is supported by the CNRS, ProFan Project. Also, We thank the Datavalor initiative at INSA Lyon for funding a part of this work.

References

[AA96] Ofer Arieli and Arnon Avron. Reasoning with logical bilattices. *Journal of Logic, Language and Information*, 5(1):25–63, 1996.

[AAS22] Munqath Al-Atar and Attila Sali. Strongly possible functional dependencies for sql. *Acta Cybernetica*, 2022.

[AGNP18] Ziawasch Abedjan, Lukasz Golab, Felix Naumann, and Thorsten Papenbrock. Data profiling. *Synthesis Lectures on Data Management*, 10(4):1–154, 2018.

[AHV95] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of databases*, volume 8. Addison-Wesley Reading, 1995.

[AL18] Giovanni Amendola and Leonid Libkin. Explainable certain answers. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 1683–1690, 2018.

[BB79] Catriel Beeri and Philip A Bernstein. Computational problems related to the design of normal form relational schemas. *ACM Transactions on Database Systems (TODS)*, 4(1):30–59, 1979.

[BB13] Leonard Bolc and Piotr Borowik. *Many-valued logics 1: theoretical foundations*. Springer Science & Business Media, 2013.

[BCFM09] Carlo Batini, Cinzia Cappiello, Chiara Francalanci, and Andrea Maurino. Methodologies for data quality assessment and improvement. *ACM computing surveys (CSUR)*, 41(3):1–52, 2009.

[BCKN18] Jaume Baixeries, Victor Codocedo, Mehdi Kaytoue, and Amedeo Napoli. Characterizing approximate-matching dependencies in formal concept analysis with pattern structures. *Discrete Applied Mathematics*, 249:18–27, 2018.

[BDFS84] Catriel Beeri, Martin Dowd, Ronald Fagin, and Richard Statman. On the structure of armstrong relations for functional dependencies. *Journal of the ACM (JACM)*, 31(1):30–46, 1984.

[Bel77] Nuel D Belnap. A useful four-valued logic. In *Modern uses of multiple-valued logic*, pages 5–37. Springer, 1977.

[Bel12] Radim Belohlavek. *Fuzzy relational systems: foundations and principles*, volume 20. Springer Science & Business Media, 2012.

[Ber11] Leopoldo Bertossi. Database repairing and consistent query answering. *Synthesis Lectures on Data Management*, 3(5):1–121, 2011.

[BFG00] Ana Burusco and Ramón Fuentes-González. Concept lattices defined from implication operators. *Fuzzy Sets and systems*, 114(3):431–436, 2000.

[BiJBN05] Jaume Baixeries i Juvillà and José Luis Balcázar Navarro. New closure operators and lattice representations for multivalued dependencies and related expressions. In *CLA 2005: proceedings of 3rd International Conference on Concept Lattices and Their Applications*, pages 22–33, 2005.

[BJFG94] Ana Burusco Juandeaburre and Ramón Fuentes-González. The study of the l-fuzzy concept lattice. *Mathware & soft computing. 1994 Vol. 1 Núm. 3 p. 209-218*, 1994.

[BKL13] Leopoldo Bertossi, Solmaz Kolahi, and Laks VS Lakshmanan. Data cleaning and query answering with matching dependencies and matching functions. *Theory of Computing Systems*, 52(3):441–482, 2013.

[BN93] B Bhuniya and P Niyogi. Lossless join property in fuzzy relational databases. *Data & knowledge engineering*, 11(2):109–124, 1993.

[BOV07] Radim Belohlavek, Stanislav Opichal, and Vilem Vychodil. Relational algebra for ranked tables with similarities: properties and implementation. In *International Symposium on Intelligent Data Analysis*, pages 140–151. Springer, 2007.

[BP82] Billy P Buckles and Frederick E Petry. A fuzzy representation of data for relational databases. *Fuzzy sets and Systems*, 7(3):213–226, 1982.

[BPU99] Patrick Bosc, Olivier Pivert, and L Ughetto. Database mining for the discovery of extended functional dependencies. In *18th International Conference of the North American Fuzzy Information Processing Society-NAFIPS (Cat. No. 99TH8397)*, pages 580–584. IEEE, 1999.

[BV05] Radim Belohlávek and Vilém Vychodil. Implications from data with fuzzy attributes vs. scaled binary attributes. In *The 14th IEEE International Conference on Fuzzy Systems, 2005. FUZZ'05.*, pages 1050–1055. IEEE, 2005.

[BV11] Radim Belohlavek and Vilem Vychodil. Codd's relational model from the point of view of fuzzy logic. *Journal of Logic and Computation*, 21(5):851–862, 2011.

[BV18] Radim Belohlavek and Vilem Vychodil. Relational similarity-based model of data part 2: dependencies in data. *International Journal of General Systems*, 47(1):1–50, 2018.

[CDP15] Loredana Caruccio, Vincenzo Deufemia, and Giuseppe Polese. Relaxed functional dependencies—a survey of approaches. *IEEE Transactions on Knowledge and Data Engineering*, 28(1):147–165, 2015.

[CGL16] Marco Console, Paolo Guagliardo, and Leonid Libkin. Approximations and refinements of certain answers via many-valued logics. In *Proceedings of the Fifteenth International Conference on Principles of Knowledge Representation and Reasoning*, pages 349–358, 2016.

[CGL19] Marco Console, Paolo Guagliardo, and Leonid Libkin. Do we need many-valued logics for incomplete information?. In *IJCAI*, pages 6141–6145, 2019.

[CKV92] G Chen, Etienne Kerre, and J Vandenbulcke. Fuzzy functional dependency and axiomatic system in a fuzzy relational data model. In *Proc. IPMU 92, Mallorca, 1992, pp. 313-316*, 1992.

[CV94] Juan C Cubero and M Amparo Vila. A new definition of fuzzy functional dependency in fuzzy relational databases. *International Journal of Intelligent Systems*, 9(5):441–448, 1994.

[Day92] Alan Day. The lattice theory of functional dependencies and normal decompositions. *IJAC*, 2(4):409–432, 1992.

[DLM92] János Demetrovics, Leonid Libkin, and Ilya B Muchnik. Functional dependencies in relational databases: A lattice point of view. *Discrete Applied Mathematics*, 40(2):155–185, 1992.

[DP02] Brian A Davey and Hilary A Priestley. *Introduction to lattices and order*. Cambridge university press, 2002.

[Gin88] Matthew L Ginsberg. Multivalued logics: A uniform approach to reasoning in artificial intelligence. *Computational intelligence*, 4(3):265–316, 1988.

[GM18] Gösta Grahne and Ali Moallemi. A useful four-valued database logic. In *Proceedings of the 22nd International Database Engineering & Applications Symposium*, pages 22–30, 2018.

[Gog67] Joseph A Goguen. L-fuzzy sets. *Journal of mathematical analysis and applications*, 18(1):145–174, 1967.

[GPW14] Sergio Greco, Fabian Pijcke, and Jef Wijsen. Certain query answering in partially consistent databases. *Proceedings of the VLDB Endowment*, 7(5):353–364, 2014.

[Grä11] George Grätzer. *Lattice theory: foundation*. Springer Science & Business Media, 2011.

[HB09] Melita Hajdinjak and Andrej Bauer. Similarity measures for relational databases. *Informatica*, 33(2), 2009.

[ILJ89] Tomasz Imieliński and Witold Lipski Jr. Incomplete information in relational databases. In *Readings in Artificial Intelligence and Databases*, pages 342–360. Elsevier, 1989.

[JCE17] Lucie Ježková, Pablo Cordero, and Manuel Enciso. Fuzzy functional dependencies: A comparative survey. *Fuzzy Sets and Systems*, 317:88–120, 2017.

[Kis90] Attila Kiss. X-decomposition of fuzzy relational databases. In *proceedings of international workshop on fuzzy sets and systems*. December Visegrad (Hungray), 1990.

[KL18] Henning Köhler and Sebastian Link. Sql schema design: foundations, normal forms, and normalization. *Information Systems*, 76:88–113, 2018.

[KLLZ16] Henning Köhler, Uwe Leck, Sebastian Link, and Xiaofang Zhou. Possible and certain keys for sql. *The VLDB Journal*, 25(4):571–596, 2016.

[KLTV06] Nick Koudas, Chen Li, Anthony KH Tung, and Rares Vernica. Relaxing join and selection queries. In *Proceedings of the 32nd international conference on Very large data bases*, pages 199–210, 2006.

[Len02] Maurizio Lenzerini. Data integration: A theoretical perspective. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 233–246, 2002.

[LGCP+21] Marie Le Guilly, Claudia Capo, Jean-Marc Petit, Vasile-Marian Scuturici, Rémi Revellin, Jocelyn Bonjour, and Gérald Cavalier. Attempt to better trust classification models: Application to the ageing of refrigerated transport vehicles. In *Intelligent Systems in Industrial Applications*, pages 15–27, Cham, 2021. Springer International Publishing.

[Lib16] Leonid Libkin. Sql's three-valued logic and certain answers. *ACM Transactions on Database Systems (TODS)*, 41(1):1–28, 2016.

[Lie82] Y Edmund Lien. On the equivalence of database models. *Journal of the ACM (JACM)*, 29(2):333–362, 1982.

[LL98] Mark Levene and George Loizou. Axiomatisation of functional dependencies in incomplete relations. *Theoretical Computer Science*, 206(1-2):283–300, 1998.

[LP19] Sebastian Link and Henri Prade. Relational database schema design for uncertain data. *Information Systems*, 84:88–110, 2019.

[LP20] Leonid Libkin and Liat Peterfreund. Handling sql nulls with two-valued logic. *arXiv preprint arXiv:2012.13198*, 2020.

[Mai83] David Maier. *The theory of relational databases*, volume 11. Computer science press Rockville, 1983.

[Mar92] George Markowsky. Primes, irreducibles and extremal lattices. *Order*, 9(3):265–290, 1992.

[MOA10] Jesús Medina and Manuel Ojeda-Aciego. Multi-adjoint t-concept lattices. *Information Sciences*, 180(5):712–725, 2010.

[MOARC09] Jesús Medina, Manuel Ojeda-Aciego, and Jorge Ruiz-Calvino. Formal concept analysis via multi-adjoint concept lattices. *Fuzzy sets and systems*, 160(2):130–144, 2009.

[MR92] Heikki Mannila and Kari-Jouko Räihä. *The design of relational databases*. Addison-Wesley Longman Publishing Co., Inc., 1992.

[Ng01] Wilfred Ng. An extension of the relational data model to incorporate ordered domains. *ACM Transactions on Database Systems (TODS)*, 26(3):344–383, 2001.

[PT84] Henri Prade and Claudette Testemale. Generalizing database relational algebra for the treatment of incomplete or uncertain information and vague queries. *Information sciences*, 34(2):115–143, 1984.

[RM88] KVSVN Raju and Arun K Majumdar. Fuzzy functional dependencies and lossless join decomposition of fuzzy relational database systems. *ACM Transactions on Database Systems (TODS)*, 13(2):129–166, 1988.

[SC11] Shaoxu Song and Lei Chen. Differential dependencies: Reasoning and discovery. *ACM Transactions on Database Systems (TODS)*, 36(3):1–41, 2011.

[SORK11] Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch. Probabilistic databases. *Synthesis lectures on data management*, 3(2):1–180, 2011.

[SPKN20] Philipp Schirmer, Thorsten Papenbrock, Ioannis Koumarelas, and Felix Naumann. Efficient discovery of matching dependencies. *ACM Transactions on Database Systems (TODS)*, 45(3):1–33, 2020.

[TSDT05] BK Tyagi, Ahmad Sharfuddin, RN Dutta, and Devendra K Tayal. A complete axiomatization of fuzzy functional dependencies using fuzzy function. *Fuzzy Sets and Systems*, 151(2):363–379, 2005.

[Wil95] Marcel Wild. Computations with finite closure systems and implications. In *International Computing and Combinatorics Conference*, pages 111–120. Springer, 1995.

[YOJ99] S Ben Yahia, Habib Ounalli, and Ali Jaoua. An extension of classical functional dependency: dynamic fuzzy functional dependency. *Information Sciences*, 119(3-4):219–234, 1999.

A Proofs of Section 4

Proposition (1)**.** *Let g be a scheme interpretation. The following properties hold:*

1. *g increasing if and only if h_{A_i} is increasing for any $1 \leq i \leq n$,*
2. *g a $\wedge$-homomorphism (resp. $\vee$-homomorphism) if and only if for any $1 \leq i \leq n$, h_{A_i} is a $\wedge$-homomorphism (resp. $\vee$-homomorphism) .*

Proof. We prove the items for increasing and $\wedge$-homomorphism. Other homomorphisms and decreasing properties are obtained in a dual way.

Item 1 First, assume that for any $1 \leq i \leq n$, h_{A_i} is increasing. Let $x = \langle x_1, \ldots, x_n \rangle$, $y = \langle y_1, \ldots, y_n \rangle \in \mathcal{L}_R$, we have

$$\begin{array}{rlll} x \leq y \iff & x_i \leq y_i & \text{for any } 1 \leq i \leq n & (\text{def of } \mathcal{L}_R) \\ \implies & h_{A_i}(x_i) \leq h_{A_i}(y_i) & \text{for any } 1 \leq i \leq n & (h_{A_i} \text{ increasing}) \\ \implies & g(x) \leq g(y) & & \end{array}$$

For the only if part, assume g is increasing and consider $x_i \leq y_i$ in $\mathcal{L}_{A_i}$. Build x and y in $\mathcal{L}_R$ as follows, $x = \langle 1_1, \ldots, x_i, \ldots, 1_n \rangle$ and $y = \langle 1_1, \ldots, y_i, \ldots, 1_n \rangle$. Since g is increasing, we have $g(x) \leq g(y)$ and for any $j \neq i$, $h_{A_j}(x_j) = h_{A_j}(y_j) = h_{A_j}(1_j) = 1$. We must also have $h_{A_i}(x_i) \leq h_{A_i}(y_i)$. Because for any pair $x_i \leq y_i$ in $\mathcal{L}_{A_i}$ and for any $1 \leq i \leq n$ we can build such x, y showing $h_{A_i}(x_i) \leq h_{A_i}(y_i)$, we have that h_{A_i} is increasing for any $1 \leq i \leq n$.

Item 2 Let us assume that for each $1 \leq i \leq n$, h_{A_i} is a $\wedge$-homomorphism. For $x, y \in \mathcal{L}_R$, one has:

$$\begin{aligned} g(x \wedge y) &= \langle h_{A_1}(x_1 \wedge y_1), \ldots, h_{A_n}(x_n \wedge y_n)\rangle \quad \text{(def of } g\text{)} \\ &= \langle h_{A_1}(x_1) \wedge h_{A_1}(y_1), \ldots, h_{A_n}(x_n) \wedge h_{A_n}(y_n)\rangle \quad (\wedge\text{-homomorphism}) \\ &= \langle h_{A_1}(x_1), \ldots, h_{A_n}(x_n)\rangle \wedge \langle h_{A_1}(y_1), \ldots, h_{A_n}(y_n)\rangle \quad (\text{in } \{0,1\}^n) \\ &= g(x) \wedge g(y) \end{aligned}$$

As for the only if part, we use contrapositive. Assume without loss of generality that h_{A_1} is not $\wedge$-homomorphic, and let $x_1, y_1 \in \mathcal{L}_{A_1}$ such that $h_{A_1}(x_1 \wedge y_1) \neq h_{A_1}(x_1) \wedge h_{A_1}(y_1)$. Let $x = \langle x_1, \ldots, x_n\rangle$ and $y = \langle y_1, \ldots, y_n\rangle$. Then, $g(x) \wedge g(y) \neq g(x \wedge y)$ by definition of g, as for A_1 we have $h_{A_1}(x_1 \wedge y_1) \neq h_{A_1}(x_1) \wedge h_{A_1}(y_1)$. □

B Proofs of Section 5

Proposition (2)**.** *Let r be a relation over a scheme context $\mathcal{C}_R$. Then, $\mathcal{L}_r \in \mathsf{Sub}^1_\wedge(\mathcal{L}_R)$.*

Proof. First, we show that $\mathcal{L}_r$ is indeed a lattice. It is sufficient to prove that it has a well-defined $\wedge$ operation and a top element [DP02]. Also following [DP02], we have that $\bigwedge \emptyset = \langle 1_{A_1}, \ldots, 1_{A_n}\rangle$, the top element of $\mathcal{L}_R$. Thus, $\mathcal{L}_r$ has a top element. Now we prove that $\mathcal{L}_r$ has a $\wedge$ operation. Since $\mathcal{L}_r \subseteq \mathcal{L}_R$, it is sufficient to prove that for every $t_1, t_2 \in \mathcal{L}_r$, $t_1 \wedge t_2 \in \mathcal{L}_r$, where $\wedge$ is the meet operation of $\mathcal{L}_R$. By definition of $\mathcal{L}_r$, $t_1 = \bigwedge T_1$ and $t_2 = \bigwedge T_2$ for some $T_1, T_2 \subseteq f_R(r)$. Since, $t_1 \wedge t_2 = \bigwedge T_1 \wedge \bigwedge T_2 = \bigwedge T_1 \cup T_2$ and $T_1 \cup T_2 \subseteq f_R(r)$, $t_1 \wedge t_2 \in \mathcal{L}_r$ holds, by construction of $\mathcal{L}_r$. Consequently, $\mathcal{L}_r$ is indeed a lattice. As $\mathcal{L}_r \subseteq \mathcal{L}_R$ and $\mathcal{L}_r$ is closed for the $\wedge$ operation, it is furthermore a $\wedge$-sublattice of $\mathcal{L}_R$. □

Proposition (6)**.** *Let r be a relation over $\mathcal{C}_R$, g a reality, and $x \rightarrow y$ an abstract FD. We have $r \models_g g(x) \rightarrow g(y)$ if and only if for any $z \in \mathcal{L}_r$, $g(x) \subseteq g(z)$ implies $g(y) \subseteq g(z)$, denoted $\mathcal{L}_r \models_g g(x) \rightarrow g(y)$.*

Proof. The if part is clear. We prove the only if part. Let $z \in \mathcal{L}_r$. If $z \in f_R(r)$ then $g(x) \subseteq g(z)$ implies $g(y) \subseteq g(z)$ by assumption. Hence consider an element $z \in \mathcal{L}_r \setminus f_R(r)$ such that $g(x) \subseteq g(z)$. By definition of $\mathcal{L}_r$, $z = \bigwedge F_z$ where $F_z = \{m \in f_R(r) \mid z \leq m\}$. As g is increasing, it must be that for all m in F_z, $g(x) \subseteq g(m)$. Since $r \models_g g(x) \rightarrow g(y)$, we also have that $g(y) \subseteq g(m)$ for any $m \in F_z$, and in particular $g(y) \subseteq \bigcap_{m \in F_z} g(m)$. Moreover, g is a $\wedge$-homomorphism so that $z = \bigwedge F_z$ implies $g(z) = g(\bigwedge F_z) = \bigcap_{m \in F_z} g(m)$. Therefore $g(y) \subseteq g(z)$ and $\mathcal{L}_r \models_g g(x) \rightarrow g(y)$. □

Lemma (1)**.** *Let r be a relation over a scheme context $\mathcal{C}_R$. Let $x \in \mathcal{L}_R$, and g be a reality. Then $g(\phi(\pi_g(x))) = \phi_g(g(\pi_g(x))) = \phi_g(g(x))$.*

Proof. The equality $\phi_g(g(\pi_g(x))) = \phi_g(g(x))$ directly follows by construction of $\pi_g(x)$ as $g(\pi_g(x)) = g(x)$. As $\pi_g(x) \leq \phi(\pi_g(x))$ in $\mathcal{L}_R$ and g is increasing, we have that $g(x) = g(\pi_g(x)) \subseteq \phi_g(g(\pi_g(x)))$. To show that $g(\phi(\pi_g(x))) = \phi_g(g(x))$, we prove that $\phi(\pi_g(x)) = \mathsf{min}_{\leq}\{y \in \mathcal{L}_r \mid g(x) \subseteq g(y)\}$, which implies that $g(\phi(\pi_g(x))) = \mathsf{min}_{\subseteq}\{F \in g(\mathcal{L}_r) \mid g(x) \subseteq F\} = \phi_g(g(x))$. If $\phi(\pi_g(x))$ is the bottom element of $\mathcal{L}_r$, $g(\phi(\pi_g(x)))$ must be the bottom element of $g(\mathcal{L}_r)$ and $g(\phi(\pi_g(x))) = \phi_g(g(x))$ is clear. If this is not the case, let $y \not\geq$

$\phi(\pi_g(x))$, $y \in \mathcal{L}_r$. By closure properties, $y \not\geq \phi(\pi_g(x))$ is equivalent to $y \not\geq \pi_g(x)$. To satisfy $y \not\geq \pi_g(x)$, there must be $A \in R$ such that $\pi_g(x)[A] \not\leq_A y[A]$. We have two cases: $A \in g(x)$ or $A \notin g(x)$. If $A \notin g(x)$, we have that $A \notin g(\pi_g(x))$ as $g(x) = g(\pi_g(x))$. Consequently, $\pi_g(x)[A] = 0_A$ by construction of $\pi_g(x)$ and hence $\pi_g(x)[A] \leq_A y[A]$. Therefore, it must be that $A \in g(x)$. As $\pi_g(x)[A] \not\leq_A y[A]$ by assumption, it follows that $A \notin g(y)$ and therefore that $g(x) \not\subseteq g(y)$. Consequently, for any $y \not\geq \pi_g(x)$, $y \in \mathcal{L}_r$, we have $g(x) \not\subseteq g(y)$ so that $g(\phi(\pi_g(x))) = \mathsf{min}_{\subseteq}\{F \in g(\mathcal{L}_r) \mid g(x) \subseteq F\} = \phi_g((x))$, concluding the proof. □

Lemma (2)**.** *Let r be a relation over $\mathcal{C}_R$, and $X \to A$ a functional dependency. Then:*

1. *$X \to A$ is certain if and only if for any $x \in \mathcal{L}_r$, either $x[A] = 1_A$ or there exists $B \in X$ such that $x[B] = 0_B$.*
2. *$X \to A$ is strongly certain if and only if for any $x \in \mathcal{L}_r$ either $x[A] \geq_A \bigvee \mathsf{CPr}(\mathcal{L}_A)$ or there exists $B \in X$ such that $x[B] \not\geq_B c$ for any $c \in \mathsf{CPr}(\mathcal{L}_B)$.*

Proof. We prove the items in order.

Item 1 We begin with the if part. There are two cases, either $X = \emptyset$, or $X \neq \emptyset$. In the first case, our assumption implies that $x[A] = 1_A$ for any $x \in \mathcal{L}_r$. Hence, by construction of a reality g, $A \in g(x)$ for any $x \in \mathcal{L}_r$, in particular the bottom element of $\mathcal{L}_r$, and $r \models_g X \to A$ holds. Assume $X \neq \emptyset$. Let $x \in \mathcal{L}_r$, and g be any reality. If $x[A] = 1_A$, then $A \in g(x)$ since $h_A(1_A)$ must be 1 by definition of an interpretation. Hence if $X \subseteq g(x)$, we still have $A \in g(x)$ and x satisfies $X \to A$ under g. Suppose now $x[A] \neq 1_A$. By assumption, there exists $B \in X$ such that $x[B] = 0_B$. Hence $h_B(x[B]) = 0$, and $B \notin g(x)$ so that $X \not\subseteq g(x)$. Therefore, in any case, x satisfies $X \to A$ through g and $r \models_g X \to A$ for any $g \in \mathcal{R}$.

We prove the only if part using contrapositive. Assume there exists $x \in \mathcal{L}_r$ such that for all $B \in X$, $x[B] \neq 0_B$ and $x[A] \neq 1_A$. We construct a reality g such that $r \not\models_g X \to A$:

- for A put $h_A(y_A) = 1$ if $y_A \geq z_A$ in $\mathcal{L}_A$ with z_A some cover of $x[A]$. Note that such a cover must exist since $x[A] \neq 1_A$. Put $h_A(y_A) = 0$ otherwise, so in particular $h_A(x[A]) = 0$.
- For any B in X, let $h_B(y_B) = 1$ if $y_B \geq x[B]$ in $\mathcal{L}_B$ and $h_B(y_B) = 0$ otherwise. Since $x[B] \neq 0_B$, $h_B(0_B) = 0$ is satisfied.
- For any other attribute $C \in R$, let $h_C(y_C) = 1$ if $y_C = 1_C$ and 0 otherwise.

It is clear that any h thus defined is a $\wedge$-homomorphism. Using Proposition 2 and Definition 6, the mapping g obtained by combining all attributes interpretations is a reality. If $X = \emptyset$, by construction of g and x, $A \notin g(x)$ and hence $r \not\models_g X \to A$. Assume $X \neq \emptyset$. For any $B \in X$, one has $B \in g(x)$ as $g_{|B}(x) = h_B(x[B]) = 1$ by definition of h_B. We also have $A \notin g(x)$ since $g_{|A}(x) = h_A(x[A]) = 0$ by construction of h_A. Therefore $r \not\models_g X \to A$, and $X \to A$ is not certain, concluding the proof of this item.

Item 2 We begin with the if part. Let g be a strong reality and let $x \in \mathcal{L}_r$. If $X \not\subseteq g(x)$, then $g(x)$ vacuously satisfies $X \to A$. Let us assume that $X \subseteq g(x)$ so that for any $B \in X$, there is $c \in \mathsf{CPr}(\mathcal{L}_B)$ such that $x[B] \geq c$. As g is a strong reality, there must be some $c' \in \mathsf{CPr}(\mathcal{L}_A)$ such that $g_{|A} = h_A^{c'}$ by Proposition 5. By assumption, we have that $x[A] \geq c'$ as $x[A] \geq \bigvee \mathsf{CPr}(\mathcal{L}_A)$. Therefore, $A \in g(x)$ and we conclude that $r \models_g X \to A$ for any

reality $g \in \mathcal{R}_s$. Note that in the case where $X = \emptyset$, $X \subseteq g(x)$ holds for any $x \in \mathcal{L}_r$ so that for all such x, we have $x[A] \geq \bigvee \mathsf{CPr}(\mathcal{L}_A)$.

We move to the only if part. Let us assume that $X \to A$ is strongly certain, and assume that there exists $x \in \mathcal{L}_r$ such that for any $B \in X$, $x[B] \geq c$ for some $c \in \mathsf{CPr}(\mathcal{L}_B)$. If $X = \emptyset$, this is the case for any $x \in \mathcal{L}_r$. We put $k = |\mathsf{CPr}(\mathcal{L}_A)|$. We consider the sequence $g_1, \ldots, g_k$ of realities as follows, for $1 \leq i \leq k$:

- $g_{i_{|B}} = h^c_B$ for some $c \in \mathsf{CPr}(\mathcal{L}_B)$ such that $x[B] \geq c$, $B \in X$,
- $g_{i_{|A}} = h^c_A$ for some $c \in \mathsf{CPr}(\mathcal{L}_A)$ such that $g_{i_{|A}} \neq g_{j_{|A}}$ for any $1 \leq j < i$.
- $g_{i_{|C}} = h^c_C$ for some $c \in \mathsf{CPr}(\mathcal{L}_C)$, $C \in R \setminus X \cup \{A\}$

For any $1 \leq i \leq k$, we have that $X \subseteq g_k(x)$. As $X \to A$ is strongly certain, we must also have $A \in g_k(x)$ by Definition 9. By construction of the sequence $g_1, \ldots, g_k$, for any $c \in \mathsf{CPr}(\mathcal{L}_A)$ there is a (unique) $1 \leq i \leq k$ such that $g_{i_{|A}} = h^c_A$. However, as $A \in g_i(x)$ for any g_i, we have that $x[A] \geq c$ for any $c \in \mathsf{CPr}(\mathcal{L}_A)$ and hence $x[A] \geq \bigvee \mathsf{CPr}(\mathcal{L}_A)$. Since this reasoning can apply to any $x \in \mathcal{L}_r$, the second item follows. □

Lemma (3)**.** *Let r be a relation over $\mathcal{C}_R$, and $X \to A$ a functional dependency. Then:*

1. *$X \to A$ is not possible if and only if there exists $x \in \mathcal{L}_r$, such that $x[A] = 0_A$ and $x[B] = 1_B$ for any $B \in X$.*
2. *$X \to A$ is strongly possible if and only if there exists $c_X \in \mathcal{L}_R$ such that $c_X[B] = c$, for some c in $\mathsf{CPr}(\mathcal{L}_B)$, for every $B \in X$, $c_X[B] = 0_B$ for every $B \notin X$, and such that $\phi(c_X)[A] \geq c$ for some $c \in \mathsf{CPr}(\mathcal{L}_A)$.*

Proof. We prove the statements in order. Let r be a relation over $\mathcal{C}_R$ and $X \to A$ a functional dependency. For the first item we need to introduce particular realities. Recall that $\mathcal{A}(\mathcal{L}_A)$ is the set of atoms of $\mathcal{L}_A$. Let $a \in \mathcal{A}(\mathcal{L}_A)$. We define a reality g^a as follows:

$$g^a_{|B}(x) = \begin{cases} 1 & \text{if } x[B] = 1_B \\ 0 & \text{otherwise} \end{cases} \quad \text{for any } B \neq A \text{ and } g^a_{|A}(x) = \begin{cases} 1 & \text{if } x[A] \geq a \\ 0 & \text{otherwise} \end{cases}$$

Item 1 We begin with the if part. Hence, assume there exists $x \in \mathcal{L}_r$ such that $x[A] = 0_A$ and for every $B \in X$, $x[B] = 1_B$. In the case where $X = \emptyset$, then x just satisfies $x[A] = 0_A$. Let g be a reality. By definition of interpretations, we must have $A \notin g(x)$ and $X \subseteq g(x)$. Hence, x does not satisfy $X \to A$, so that $r \not\models_g X \to A$. Since this holds for any reality g, we conclude that $X \to A$ is not possible.

We show the only if part. Assume that $X \to A$ is not possible in $\mathcal{L}_r$, that is for any reality g, $\mathcal{L}_r \not\models_g X \to A$. In particular, this is true for any reality g^a, $a \in \mathcal{A}(\mathcal{L}_A)$. Then, for any $a \in \mathcal{A}(\mathcal{L}_A)$, there exists $x_a \in \mathcal{L}_r$ such that $x[B] = 1_B$ for any $B \in X$ and $x[A] \not\geq_A a$, by definition of g^a and 9. As $\mathcal{L}_r$ is a meet-sublattice of $\mathcal{L}_R$, $x = \bigwedge_{a \in \mathcal{A}(\mathcal{L}_A)} x_a$ also belongs to $\mathcal{L}_r$. However, it must be that $x[B] = 1_B$ for any $B \in X$. Furthermore, we have that $x[A] = 0_A$ as $x_a \not\geq_A a$ for any $a \in \mathcal{A}(\mathcal{L}_A)$, which concludes the proof of the item.

Item 2 We begin with the only if part. Let g be a strong reality such that $r \models_g X \to A$ and consider the following c_X:

- $c_X[B] = 0_B$ for $B \notin X$,

- $c_X[B] = c$ for $c \in \mathsf{CPr}(\mathcal{L}_B)$ such that $g_{|B} = h^c_B$ for any $B \in X$. Such a c exists by Proposition 5 as g is a strong reality.

Again, if $X = \emptyset$, then $c_X[B] = 0_B$ for every $B \in R$. We consider $\phi(c_X) \in \mathcal{L}_r$. As $c_X \leq \phi(c_X)$, and by construction of c_X, we must have $X \subseteq g(\phi(c_X))$. However, $r \models_g X \to A$ which implies that $A \in g(\phi(c_X))$ by Definition 9. Hence, since g is a strong reality, there must be some $c \in \mathsf{CPr}(\mathcal{L}_A)$ such that $g_{|A} = h^c_A$ and $\phi(c_X)[A] \geq c$.

We move to the if part. Assume such c_X exists. We construct a strong reality g such that $r \models_g X \to A$ as follows:

- $g_{|B} = h^c_B$ for some $c \in \mathsf{CPr}(\mathcal{L}_B)$ if $B \notin X \cup \{A\}$,
- $g_{|B} = h^{c_X[B]}_B$ if $B \in X$,
- $g_{|A} = h^c_A$ for some $c \in \mathsf{CPr}(\mathcal{L}_A)$ such that $c \leq \phi(c_X)[A]$. Note that such c must exist by assumption.

Thus defined, g is a strong reality as $c_X[B] \in \mathsf{CPr}(\mathcal{L}_B)$ for any $B \in X$. Now we show that $r \models_g X \to A$. Let $x \in \mathcal{L}_r$. Either $x \geq \phi(c_X)$ or $x \ngeq \phi(c_X)$. In the first case, by construction of g we have $X \subseteq g(x)$, but we also have $A \in g(x)$ as $g_{|A} = h^c_A$ with $c \leq \phi(c_X)[A]$, so that $X \to A$ is satisfied for $g(x)$. Now suppose $x \ngeq \phi(c_X)$. We have that $x \ngeq c_X$. However for any $B \notin X$, $c_X[B] = 0_B$ so that necessarily $c_X[B] \leq x[B]$. Therefore, there must exist $B \in X$ such that $x[B] \ngeq c_X[B]$. Hence, by construction of g we will obtain $B \notin g(x)$ and consequently $X \nsubseteq g(x)$ so that $X \to A$ is also satisfied by $g(x)$. As for any $x \in \mathcal{L}_r$, $X \subseteq g(x)$ implies $A \in g(x)$, we conclude that $r \models_g X \to A$ and $X \to A$ is strongly possible. □

C Computational results on abstract FDs

In this section, we detail some computational characteristics of abstract FDs. First, we recall some notations to ease the understanding. Let $\mathcal{C}_R$ be a scheme context with comparability function f_R and corresponding lattice $\mathcal{L}_R$ of abstract tuples. Let $\mathcal{L} \in \mathsf{Sub}^1_\wedge(\mathcal{L}_R)$ with associated closure operator ϕ and let $x \to y$ be an abstract FD. We write $\mathcal{L} \models x \to y$ if for every $z \in \mathcal{L}$, $x \leq z$ implies $y \leq z$, that is $z \models x \to y$. We have $\mathcal{L} \models x \to y$ if and only if $y \leq \phi(x)$. If Σ is a set of abstract FDs, $\mathcal{L} \models \Sigma$ if $\mathcal{L} \models x \to y$ for every $x \to y$ in Σ.

Let r a relation over $\mathcal{C}_R$. The relation r is associated to a lattice $\mathcal{L}_r \in \mathsf{Sub}^1_\wedge(\mathcal{L}_R)$ given by

$$\mathcal{L}_r = \left(\left\{\bigwedge T \mid T \subseteq f_R(r)\right\}, \leq_R\right)$$

The closure operator associated to $\mathcal{L}_r$ is ϕ_r. We write $r \models x \to y$ if for every $z \in f_R(r)$, $z \models x \to y$. The notation $r \models \Sigma$ follows. Recall from Proposition 3 in the manuscript that $r \models x \to y$ if and only if $\mathcal{L}_r \models x \to y$.

Similarly to r and $\mathcal{L}_r$, a set Σ of abstract FDs is associated to a lattice $\mathcal{L}_\Sigma$ defined by $\mathcal{L}_\Sigma = (\{x \in \mathcal{L}_R \mid x \models \Sigma\}, \leq_R)$. We call the corresponding closure operator ϕ_Σ.

We study the implication problem for abstract functional dependencies: *"given a Σ, does the abstract FD $x \to y$ follow from Σ?"* On this purpose, we use the extended Armstrong axioms for lattice implications [Day92]. They are straightforward extensions of Armstrong axioms for functional dependencies. They read as follows, for every $x, y, z \in \mathcal{L}_R$:

1. if $x \leq y$, then $\Sigma \vdash_{\mathcal{C}_R} x \to y$ (reflexivity);

2. if $\Sigma \vdash_{\mathcal{C}_R} x \to y$ then $\Sigma \vdash_{\mathcal{C}_R} x \vee z \to y \vee z$ (augmentation);

3. if $\Sigma \vdash_{\mathcal{C}_R} x \to y$ and $\Sigma \vdash_{\mathcal{C}_R} y \to z$ then $\Sigma \vdash_{\mathcal{C}_R} x \to z$ (transitivity).

Definition 12. $\Sigma \vdash_{\mathcal{C}_R} x \to y$ if there exists a derivation (or a proof) of $x \to y$ by using the extended Armstrong axioms.

Definition 13. $\Sigma \models_{\mathcal{C}_R} x \to y$ if for every relation r, $r \models_{\mathcal{C}_R} \Sigma$ entails $r \models_{\mathcal{C}_R} x \to y$.

If no confusion arises, we drop the subscript $\mathcal{C}_R$. The next propositions establish useful properties for subsequent discussions.

Proposition 10. *We have $r \models \Sigma$ if and only if $\mathcal{L}_r \subseteq \mathcal{L}_\Sigma$.*

Proof. We begin with the only if part. Assume that $r \models \Sigma$. Every abstract tuple $x \in f_R(r)$ satisfies Σ, so that $x \in \mathcal{L}_\Sigma$ holds by definition of $\mathcal{L}_\Sigma$. Since $\mathcal{L}_r = (\{\bigwedge T \mid T \subseteq f_R(r)\}, \leq_R)$, $\mathcal{L}_\Sigma \in \mathsf{Sub}^1_\wedge(\mathcal{L}_R)$ and $f_R(r) \subseteq \mathcal{L}_\Sigma$, $\mathcal{L}_r \subseteq \mathcal{L}_\Sigma$ follows.

We move to the if part. We use contrapositive. Assume that $r \not\models \Sigma$. There exists an abstract tuple $x \in f_R(r)$ such that $x \not\models \Sigma$. Thus, $x \notin \mathcal{L}_\Sigma$ and $\mathcal{L}_r \not\subseteq \mathcal{L}_\Sigma$ follows, which concludes the proof. □

The following proposition is a direct consequence of Theorem 3.1 and Theorem 3.2 in [Day92].

Proposition 11. *We have $\Sigma \vdash x \to y$ if and only if $\mathcal{L}_\Sigma \models x \to y$.*

Thus, there is a strong relationship between a set of abstract FDs and its models in the lattice $\mathcal{L}_R$. We show that deciding whether $\Sigma \vdash x \to y$ can be decided in polynomial time, using Proposition 11 and Algorithm 1. Recall that the scheme context $\mathcal{C}_R$, including each abstract lattice, is part of our input. First, we prove that Algorithm 1 correctly computes $\phi_\Sigma(x)$ for every $x \in \mathcal{L}_R$ in polynomial time.

Theorem 7. *Let $\mathcal{C}_R$ be a scheme context, Σ a set of abstract FDs and x an abstract tuple. Then, Algorithm 1 computes $\phi_\Sigma(x)$ in polynomial time in the size of Σ and $\mathcal{C}_R$.*

Proof. First we show the correctness of the algorithm. Observe that Algorithm 1 always terminates. Let $x \in \mathcal{L}_R$ and consider the output x^Σ of the algorithm. We show that $x^\Sigma = \phi_\Sigma(x)$. We first prove that x^Σ is a model of Σ and that $\phi_\Sigma(x) \leq x^\Sigma$. Suppose for contradiction that $x^\Sigma \not\models \Sigma$. Then, there exists an abstract FD $y \to z$ in Σ such that $y \leq x^\Sigma$ but $z \not\leq x^\Sigma$. However, this contradicts the fact that Algorithm 1 ends on x^Σ. Thus, $x^\Sigma \models \Sigma$ and since $x \leq x^\Sigma$, we deduce that $\phi_\Sigma(x) \leq \phi_\Sigma(x^\Sigma) = x^\Sigma$.

Now we show that $x^\Sigma \leq \phi_\Sigma(x)$. We proceed by induction on the while condition of the algorithm. At step 0, $x^\Sigma = x$ and $x^\Sigma \leq \phi_\Sigma(x)$ follows from the extensivity of ϕ_Σ. let us assume that up to a given step i, $x^\Sigma \leq \phi_\Sigma(x)$ holds. If there is no abstract FD $y \to z$ such that $y \leq x^\Sigma$ then the result follows. Now, suppose that there is an abstract FD $y \to z$ in Σ such that $y \leq x^\Sigma$. Since $y \to z$ holds in Σ, we derive that $z \leq \phi_\Sigma(y)$. Thus, we have $z \leq \phi_\Sigma(y) \leq \phi_\Sigma(x^\Sigma) \leq \phi_\Sigma(x)$ by inductive hypothesis, and using the properties of ϕ_Σ. Consequently, $z \vee x^\Sigma \leq \phi_\Sigma(x)$ holds. Hence, we derive by induction that the output x^Σ of Algorithm 1 satisfies $x^\Sigma \leq \phi_\Sigma(x)$. Consequently, $x^\Sigma = \phi_\Sigma(x)$ indeed holds.

The total complexity of Algorithm 1, depends on the cost of checking conditions of the while loop and the cost of the join operation. Since the abstract lattices of each attribute context are part of the input, this operation can be done in polynomial time in the size of Σ and $\mathcal{C}_R$, concluding the proof. □

Algorithm 1: Closure algorithm

Input: A set Σ of abstract FDs over $\mathcal{C}_R$, $u \in \mathcal{L}_R$
Output: The closure x^Σ of x.
begin
  $x^\Sigma = x$;
  while *there exists* $y \to z \in \Sigma$ *such that* $z \leq x^\Sigma$ **do**
    $x^\Sigma = x^\Sigma \vee z$;
    $\Sigma = \Sigma \setminus \{y \to z\}$;
return x^Σ ;

Remark that Algorithm 1 is a straightforward extension of classical closure algorithms [BB79]. According to Proposition 11, $\Sigma \vdash x \to y$ can be decided by computing $\phi_\Sigma(x)$ with Algorithm 1, and then testing that $y \leq \phi_\Sigma(x)$. We deduce the following theorem.

Theorem 8. *Let $\mathcal{C}_R$ be a scheme context, Σ a set of abstract FDs and $x \to y$ an abstract FD. Testing that $\Sigma \vdash x \to y$ can be done in polynomial time in the size of Σ and $\mathcal{C}_R$.*

Then, we demonstrate that $\Sigma \vdash x \to y$ (equivalently that $y \leq \phi_\Sigma(x)$), is not equivalent to $\Sigma \models x \to y$. More precisely, we show that extended Armstrong axioms are sound but incomplete. This is because comparability functions are not required to be reflexive.

Lemma 4. *Let $\mathcal{C}_R$ be a scheme context, Σ a set of abstract FDs over $\mathcal{C}_R$ and $x \to y$ another abstract FD. Then, $\Sigma \vdash x \to y$ implies $\Sigma \models x \to y$.*

Proof. Let $\mathcal{C}_R$ be a scheme context, Σ a set of abstract FDs and $x \to y$ an abstract FD such that $\Sigma \vdash x \to y$. From Proposition 11, we have $\mathcal{L}_\Sigma \models x \to y$. Let r be a relation over $\mathcal{C}_R$ such that $r \models \Sigma$. By Proposition 10, $\mathcal{L}_r \subseteq \mathcal{L}_\Sigma$ holds. As $\mathcal{L}_r \subseteq \mathcal{L}_\Sigma$, $\mathcal{L}_r \models x \to y$ holds and so does $r \models x \to y$, which concludes the proof. □

Lemma 5. *There exists a scheme context $\mathcal{C}_R$, an abstract FD $x \to y$ and a set Σ of abstract FDs such that $\Sigma \models x \to y$ but $\Sigma \not\vdash x \to y$.*

Proof. Let $R = \{A, B\}$ be a relation scheme where $\mathsf{dom}(A) = \mathsf{dom}(B) = \mathbb{N} \cup \{\mathtt{null}\}$. We associate the abstract lattice $\mathcal{L}_A$ and $\mathcal{L}_B$ presented in Figure 13 to A and B (resp.). The lattice $\mathcal{L}_R = \mathcal{L}_A \times \mathcal{L}_B$ is the lattice of all possible abstract tuples. We also define two comparability functions f_A, f_B for A, B (resp.) as follows:

$$f_A(u,v) = f_B(u,v) = \begin{cases} 1 & \text{if } u = v \text{ with } u \neq \mathtt{null} \text{ or } v \neq \mathtt{null} \\ u & \text{if } u = v = \mathtt{null} \\ 0 & \text{otherwise.} \end{cases}$$

We consider the scheme context $\mathcal{C}_R = \{(A, f_A, \mathcal{L}_A), (B, f_B, \mathcal{L}_B)\}$. Let $\Sigma = \{\langle 0,1\rangle \to \langle u,1\rangle, \langle 1,0\rangle \to \langle 1,1\rangle\}$ be a set of abstract FDs. The abstract lattice $\mathcal{L}_\Sigma$ associated to Σ is given in Figure 14. Using Proposition 11 and the fact that $\mathcal{L}_\Sigma \models x \to y$ if and only if $y \leq \phi_\Sigma$, we derive that $\Sigma \not\vdash \langle 0,u\rangle \to \langle u,0\rangle$ as $\langle 0,u\rangle \in \mathcal{L}_\Sigma$.

We show that there is no relation r over $\mathcal{C}_R$ satisfying $r \models \Sigma$ and $r \not\models \langle 0,u\rangle \to \langle u,0\rangle$. It follows that $\Sigma \models \langle 0,u\rangle \to \langle u,0\rangle$. Assume for contradiction such a r exists. By Proposition 10, $r \models \Sigma$ is equivalent to $\mathcal{L}_r \subseteq \mathcal{L}_\Sigma$. Thus, $r \not\models \langle 0,u\rangle \to \langle u,0\rangle$ implies that $\langle 0,u\rangle \in \mathcal{L}_r$ since it is the unique counter-example to $\langle 0,u\rangle \to \langle u,0\rangle$ in $\mathcal{L}_\Sigma$. Moreover, $\langle u,0\rangle$ is a meet-irreducible element of $\mathcal{L}_\Sigma$. It follows that $\langle u,0\rangle \in f_R(r)$ must hold. By definition of f_A, f_B,

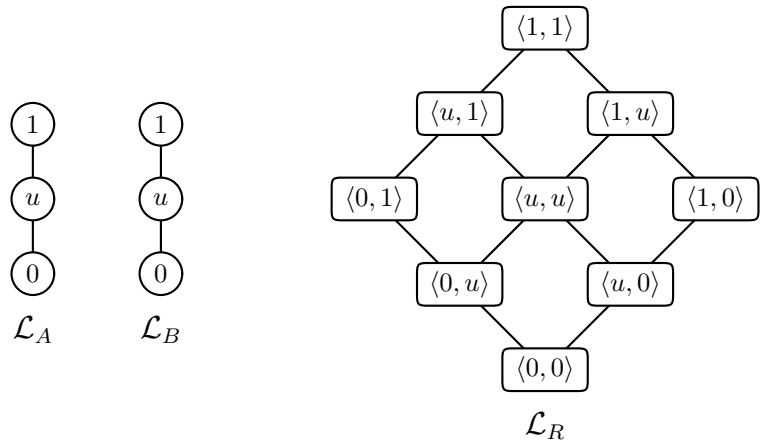


Figure 13: Abstract lattices

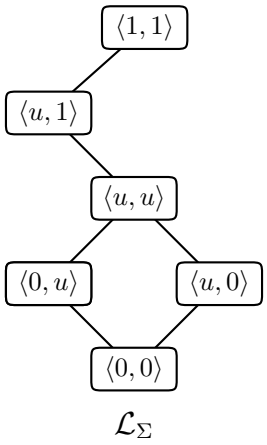


Figure 14: Abstract lattice $\mathcal{L}_\Sigma$

we deduce that r contains at least two tuples t_1, t_2 such that $t_1(A) = i$, $t_2(A) = j$ and $t_1(B) = t_2(B) = \texttt{null}$ where i, j are distinct integers. We have $f_R(t_1, t_2) = \langle 0, u\rangle$ as required. However, we also have $f_R(t_1, t_1) = \langle 1, u\rangle$ with $\langle 1, u\rangle \notin \mathcal{L}_\Sigma$. As a consequence, $\mathcal{L}_r \not\subseteq \mathcal{L}_\Sigma$ from which $r \not\models \Sigma$ follows by Proposition 10, a contradiction. □

Using Lemma 4 and Lemma 5, we obtain the following theorem.

Theorem 9. *The extended Armstrong axioms are sound but not complete.*

As a consequence, we have yet no guarantees on the complexity of deciding $\Sigma \models x \to y$. However, enforcing reflexivity for each comparability functions solves the problem.

Lemma 6. *Let $\mathcal{C}_R$ be a scheme context where for each attribute A in R, $f_A(u, u) = 1_A$ for every $u \in \mathsf{dom}(A)$. Let Σ a set of abstract FDs over $\mathcal{C}_R$ and $x \to y$ another abstract FD. Then, $\Sigma \models x \to y$ implies $\Sigma \vdash x \to y$.*

Proof. We show the contrapositive. Assume that $\Sigma \not\vdash x \to y$. We construct a relation r such that $r \models \Sigma$ but $r \not\models x \to y$. Using Proposition 11, $\Sigma \not\vdash x \to y$ is equivalent to $y \not\leq \phi_\Sigma(x)$. Thus, let $r = \{t_1, t_2\}$ be a relation over $\mathcal{C}_R$ such that $f_R(t_1, t_2) = \phi_\Sigma(x)$. Such a relation must exist as comparability functions are surjective by definition. Now, as each comparability function is assumed to be reflexive, we have $f_R(t_1, t_1) = f_R(t_2, t_2) = 1_R$ where 1_R is the top element of $\mathcal{L}_R$. Because $\mathcal{L}_\Sigma \in \mathsf{Sub}^1_\wedge(\mathcal{L}_R)$ and $\phi_\Sigma(x) \in \mathcal{L}_\Sigma$, we deduce

that $\mathcal{L}_r \subseteq \mathcal{L}_\Sigma$. By Proposition 10, this is equivalent to $r \models \Sigma$. Now since $y \not\leq \phi_\Sigma(x)$ with $\phi_\Sigma(x) \in \mathcal{L}_r$, we deduce that $r \not\models x \to y$, which concludes the proof. □

Theorem 10. *The extended Armstrong axioms are sound and complete provided comparability functions are reflexive.*

In spite of Theorem 9, the work of Day [Day92] on lattice implications still allows to reason on abstract FDs independently of any relations on $\mathcal{C}_R$. Let Σ, Σ' be two sets of abstract FDs. We say that Σ, Σ' are *equivalent* if $\mathcal{L}_\Sigma = \mathcal{L}_{\Sigma'}$. Putting $\Sigma \vdash \Sigma'$ if $\Sigma \vdash x \to y$ for every $x \to y \in \Sigma'$, we immediately derive:

Proposition 12. *The sets of abstract FDs Σ_1 and Σ_2 are equivalent if and only if $\Sigma_1 \vdash \Sigma_2$ and $\Sigma_2 \vdash \Sigma_1$. Moreover, $\Sigma_1 \vdash \Sigma_2$ can be tested in polynomial time in the size of Σ_1, Σ_2 and $\mathcal{C}_R$.*

A set Σ of abstract FDs is *redundant* if there exists an abstract FD $x \to y$ in Σ such that $\Sigma \setminus \{x \to y\} \vdash x \to y$. It is *irredundant* otherwise. An abstract FD $x \to y$ is *right-closed* in Σ if $y = \phi_\Sigma(x)$. The *left-saturation* of an abstract FD $x \to y$ in Σ is the abstract FD $\phi_{\Sigma'}(x) \to y$ where $\Sigma' = \Sigma \setminus \{x \to y\}$. Eventually, Σ is *minimal* if for every equivalent set of abstract FDs Σ', $|\Sigma| \leq |\Sigma'|$, where $|\Sigma|$ is the number of abstract FDs in Σ. The Theorem 6.1 of Day [Day92] can be restated and adapted to our terms as follows:

Theorem 11 ([Day92])**.** *Let $\mathcal{C}_R$ be a scheme context and Σ be a set of abstract FD. Consider the following three-step algorithm applied to Σ and let Σ_m be the resulting set of abstract FDs:*

- *right-closure*
- *left-saturation*
- *remove redundancy*

Then, Σ_m is minimal.

Observe that Theorem 11 is the lattice counterpart of theorems on minimal covers for functional dependencies [Mai83, Wil95]. With the help of Proposition 11, Algorithm 1, and Theorem 11, we obtain

Theorem 12. *Let $\mathcal{C}_R$ be a scheme context, Σ a set of abstract FDs over $\mathcal{C}_R$. The following tasks can be computed in poylnomial time in the size of $\mathcal{C}_R$ and Σ:*

- *computing an equivalent irredundant set Σ' of abstract FDs,*
- *computing an equivalent minimal set Σ of abstract FDs.*

We conclude this section by giving hardness results regarding abstract FDs. On this purpose, we show how abstract FDs generalize classical FDs. Beforehand, we define some more notions. A set of abstract FDs Σ is an *abstract cover* for a relation r if for every abstract FD $x \to y$, $r \models x \to y$ if and only if $\Sigma \models x \to y$. An abstract tuple x is an *abstract key* of r if $\phi_r(x) = 1_R$, ϕ_r being the closure operator associated to r. The abstract key x is minimal if for every $y < x$ in $\mathcal{L}_r$, y is not an abstract key of r. The size of an abstract key is its number of non-bottom elements.

The subsequent discussion is a more formal version of the end of the paragraph *Relational model (without nulls)* of Subsection 4.2. in the revised manuscript. Let R be a

relation scheme such that $|R| = n$, and $\texttt{null} \notin \mathsf{dom}(A)$ for each $A \in R$. Let $\mathbf{F}$ be a set of functional dependencies over R. For each $A \in R$, let $\mathcal{L}_A$ be the lattice $0 \leq 1$ and let f_A be the comparability function given by $f(u, v) = 1$ if and only if $u = v$. Let $\mathcal{C}_A = (A, f_A, \mathcal{L}_A)$ be the resulting attribute context and let $\mathcal{C}_R = \{\mathcal{C}_A \mid A \in R\}$ be the associated scheme context. An abstract tuple x is then a binary word over $\{0, 1\}^n$, which can be interpreted as the characteristic vector of the corresponding subset X of R. The context $\mathcal{C}_R$ admits a unique (strong) reality defined by $x_g = 1_R$, where 1_R is the top element of $\mathcal{L}_R$. The corresponding reality g precisely coincides with the classical equality. On the other hand, since the `null` value is not considered, the comparability functions are reflexive. Thus, Theorem 9 applies and $\Sigma \models x \to y$ can be checked in polynomial time for every set of abstract FDs Σ and every abstract FD $x \to y$. This lays the ground for the study of classical problems regarding abstract keys and abstract covers of a relation.

The following proposition settles the relationship between abstract FDs and FDs in this context.

Proposition 13. *We have $r \models X \to Y$ if and only if $r \models x \to y$. As a consequence, we have:*

- *an abstract tuple x is an abstract minimal key of r if and only if the corresponding set X is a minimal key of r,*
- *a set of abstract FDs Σ is an abstract cover of r if and only if the corresponding set of FDs $\mathbf{F}$ is a cover of r.*

Proof. This is a consequence of Theorem 2 in the paper and the fact that the unique reality g coincide with strict equality. □

We derive the following more general theorem, which applies to any scheme context.

Theorem 13. *Let $\mathcal{C}_R$ be any scheme context. Then:*

- *the problem of computing the size of a minimal abstract key of a relation is NP-complete,*
- *the problem of listing the abstract minimal keys of a relation is harder than the problem of listing the minimal keys of a relation.*
- *the problem of computing an abstract minimal cover of a relation is harder than the problem of computing a minimal cover of a relation.*