Dependency Covers from a FCA Perspective

Jaume Baixeries[1], Victor Codocedo[2], Mehdi Kaytoue[3] and Amedeo Napoli[4]

[1]*Universitat Politècnica de Catalunya, Barcelona, Catalonia*
[2]*Instituto para la Resiliencia ante Desastres, Chile*
[3]*Université de Lyon. CNRS, INSA-Lyon, LIRIS, Lyon, France*
[4]*Université de Lorraine, CNRS, LORIA, Nancy, France*

Abstract
Implications in Formal Concept Analysis (FCA), Horn clauses in Logic, and Functional Dependencies (FDs) in the Relational Database Model, are very important dependency types in their respective fields. Moreover, they have been proved to be equivalent from a syntactical point of view. Then notions and algorithms related to one dependency type in a field can be reused and applied to another dependency type in the other field. One of these notions is that of *cover*, also known as a *base* or *basis*, i.e., a compact representation of a complete set of implications, FDs, or Horn clauses. Although the notion of cover exists in the three fields, the characterization and the related uses of a cover are different. In this paper, we study and compare, from an FCA perspective, the principles on which rely the most important covers in each field. Finally, we discuss some open questions that are of interest in the three fields, and especially to the FCA community.

Keywords
Functional dependencies, Implications, Horn Clauses, Dependency Covers, Closure

1. Introduction and Motivation

A **dependency** is a relation between sets of attributes in a dataset. In this paper, they are represented as $X \to Y$, where the type of the subsets of attributes X and Y, and the semantics of $\to$ may vary w.r.t. the context. There are many different kinds of dependencies: complete and comprehensive surveys, from a Relational Database Theory perspective, can be found in [1] and in [2]. Here, we focus on those dependencies that follow the so called Armstrong axioms, this is, reflexivity, augmentation and transitivity, which appear in different fields of computer science: *functional dependencies* (FDs) in the Relational Database Model, *Horn clauses* in logic and logic programming, and *implications* in Formal Concept Analysis.

Functional dependencies [3] are of paramount importance in the Relational Database Model (RDBM), where they are used to express constraints or rules that need to hold in a database, to help the design of a database or to check for errors and inconsistencies. A set of

Published in Sergei O. Kuznetsov, Amedeo Napoli, Sebastian Rudolph (Eds.): The 11th International Workshop "What can FCA do for Artificial Intelligence?", FCA4AI 2023, co-located with IJCAI 2023, August 20 2023, Macao, S.A.R. China, Proceedings, pp. 57–66.
jbaixer@cs.upc.edu (J. Baixeries); victor.codocedo@gmail.com (V. Codocedo); mehdi.kaytoue@gmail.com (M. Kaytoue); amedeo.napoli@loria.fr (A. Napoli)
0000-0003-2896-2247 (J. Baixeries); 0000-0002-6360-2039 (V. Codocedo); 0000-0002-1569-5242 (M. Kaytoue); 0000-0001-5236-9561 (A. Napoli)

Horn clauses [4] is a special case of Boolean functions that are crucial in logic programming [5, 6] and artificial intelligence (see [7] for a detailed explanation). **Implications** are at the core of Formal Concept Analysis (FCA) where they are used to model and deduce relevant information that is contained in a formal context [8, 9].

Although they appear in different fields, these three constructions have been applied on different kinds of data and have been used for different purposes. They also all share the same axioms, which means that, from a syntactical point of view, they are all equivalent. More specifically, the equivalence between functional dependencies and Horn clauses is presented in [10, 11] (see also [7] for a more detailed explanation). The equivalence between functional dependencies and implications is explained in [8] and the equivalence between implications and Horn clauses is explained in Section 5.1 in [9] as well as in [12]. These equivalences allow us to talk in a generic way of **Armstrong dependencies** or, simply, **dependencies**.

One of the consequences of this equivalence is the transversality of concepts, problems and algorithms between these three fields. One of the most typical examples is the decision problem of the **logical implication** which consists in, given a set of dependencies Σ and a single dependency σ, to determine whether σ is logically entailed by Σ, that is, $\Sigma \models \sigma$. Entailment means that σ can be derived from Σ by the iterative application of the Armstrong axioms. This problem is named **implication problem** in the RDBM [13, 14] and FCA fields, and **deduction** in logic [7]. It is of capital interest in all three fields. In the RDBM it allows to test whether two different sets of functional dependencies are equivalent [2], and it also allows to compute a more succinct set of functional dependencies, which is relevant to assist the design process of databases [15, 16]. In logic the deduction problem is used to check whether a logical expression is consistent w.r.t. a knowledge base and to compute the prime implicants of a Horn function [7]. In Formal Concept Analysis this problem is used, for instance, in attribute exploration [9], which consists in creating a data table (context) in agreement w.r.t. a set of attributes and a set of objects, and also for computing the Duquenne-Guigues basis [17].

Roughly speaking, the computation of the logical implication problem $\Sigma \models X \rightarrow Y$ is performed by iterating over Σ and applying the Armstrong axioms to infer new dependencies until a positive or negative answer is found. However, this problem can be reduced to the computation of the **closure** of X with respect to Σ ($\mathrm{closure}_\Sigma(X)$). This closure returns the largest set such that $\Sigma \models X \rightarrow \mathrm{closure}_\Sigma(X)$ holds. Therefore, the implication problem $\Sigma \models X \rightarrow Y$ boils down to testing whether $Y \subseteq \mathrm{closure}_\Sigma(X)$.

As an example of transversality, the algorithm that computes $\mathrm{closure}_\Sigma(X)$ appears in most of the main database textbooks, where it is called **closure** [14, 13, 3], and also in logic, where it is called **forward chaining** [7], while in FCA the same algorithm that first appeared in the RDBM is discussed and reused in [9].

Another "transversal notion" which is present in all three fields is the notion of **cover**. In general terms, it is not suitable to handle the set Σ of all dependencies that hold, because of its potential large size, but rather a subset of Σ that contains the same information and that is significantly smaller in size. By *"containing the same information"* we mean that this subset may generate, thanks to the application of the Armstrong axioms, the complete set Σ. This compact and representative subset is called "cover" in the RDBM, "basis" in FCA and "set of prime implicants" in logic. Moreover, each field has defined and used a different kind of cover. While in the RDBM this base is the Canonical-Direct Basis or the Minimal Cover, the cover of choice in FCA is the so-called Duquenne-Guigues basis.

Both the implication problem and the problem of computing a cover are related: the implication problem is used in the algorithm **Canonical Basis** (Algorithm 16, page 103 in [9]) to compute the Duquenne-Guigues basis, and it is also used in the algorithm **Direct** (Section 5.4, Chapter 5 in [13]) which is used to compute the Minimal Cover. Again, the transversality of the Armstrong dependencies appears in a general concept (computing a cover) but in different forms (Duquenne-Guigues basis and Minimal Cover).

This paper is a short version of the paper *Three Views on Dependency Covers from an FCA Perspective* that was accepted at the ICFCA 2023 (International Conference on Formal Concept Analysis). The purpose of this study is to present a discussion of the main different covers used in the RDBM, in Logic, and in FCA from the perspective of the formal concept analysis community. To do so, we review three different main covers that appeared in the literature.

The paper is organized as follows. Section 2 provides the necessary definitions needed in the paper. Section 3 includes a detailed comparison of the main covers. Finally, Section 4 concludes the paper and proposes an extensive discussion about these different and important covers.

2. Definitions

In this section we introduce the definitions used in this paper. We do not provide the references for all of them because they can be found in all the textbooks and papers related to the RDBM, Logic and FCA.

As explained in the introduction, implications[8], functional dependencies [13] and Horn clauses [4] are dependencies between sets of attributes, which are equivalent from a syntactical point of view, since they are in agreement with the Armstrong axioms.

Definition 2.1. *Given set of attributes $\mathcal{U}$, for any $X, Y, Z \subseteq \mathcal{U}$, the Armstrong axioms are:*

1. ***Reflexivity**: If $Y \subseteq X$, then $X \to Y$ holds.*
2. ***Augmentation**. If $X \to Y$ holds, then $XZ \to YZ$ holds.*
3. ***Transitivity**. If $X \to Y$ and $Y \to Z$ hold, then $X \to Z$ holds.*

When we write that a dependency $X \to Y$ **holds**, we mean all the instances in which this dependency is valid or true. Therefore, the sentence *"If $X \to Y$ holds, then $XZ \to YZ$ holds"* can be rephrased as *"In any instance in which $X \to Y$ is valid, the dependency $XZ \to YZ$ is valid as well".*

The Armstrong axioms allow us to define the closure of a set of dependencies as the iterative application of these axioms over a set of dependencies.

Definition 2.2. Σ^+ *denotes the closure of a set of dependencies Σ, and can be constructed thanks to the iterative application of the Armstrong axioms over Σ. This iterative application terminates when no new dependency can be added, and it is finite. Therefore, Σ^+ contains the largest set of dependencies that hold in all instances in which all the dependencies in Σ hold.*

The closure of a set of dependencies induces the definition of the cover of such a set of dependencies.

Definition 2.3. *The **cover** or **basis** of a set of dependencies Σ is any set Σ' such that $\Sigma'^+ = \Sigma^+$.*

3. Covers of Dependencies

We now present the different types of covers that are present and adopted in the three fields, namely RDBM, FCA, and Logic. Since the definition of a cover (Definition 2.3) is very generic, the covers reviewed here have been defined with respect to specific characteristics and for different purposes.

3.1. Four Main Characteristics of Covers

In general terms, a cover is simply a set of dependencies Σ. The definition of a cover is very generic and below we introduce some relevant properties which are useful for characterizing the different covers. In particular, Σ is equivalent to another set of dependencies Σ' modulo the Armstrong axioms when the closures Σ^+ and Σ'^+ are the same.

There are three sources of redundancy in a set of dependencies: reflexivity, augmentation and transitivity (with respect to the Armstrong axioms 2.1), but here dependencies that can be derived by reflexivity 2.1 are **trivial** and not considered in the following discussion. This assumption does not invalidate any of the arguments that are to be presented, but simplifies the discussion. We will present three bases that try to reduce the number of dependencies that are needed in order to compute the closure $\mathrm{closure}_\Sigma$ for any set of attributes. But first, we need to characterize the said bases according to the different ways to remove redundancy that are used.

Definition 3.1. *A set of dependencies Σ is **left-reduced** if and only if for all $X \to Y \in \Sigma$ there is no $X' \to Y$, where $X' \subset X$, such that changing $X \to Y$ by $X' \to Y$ in Σ gives an equivalent base.*

The process of left-reducing a set of dependencies is also mentioned as the removal of **extraneous attributes** in the left-hand sides of all dependencies. As it can be expected, an attribute x is extraneous in the left-hand side of a dependency $\sigma \in \Sigma$ if removing x from the left-hand side in σ does not change Σ^+.

Example 3.1. *Let us take the set of dependencies $\Sigma = \{\, a \to b, b \to ac, a \to c, bc \to a \,\}$. In this set, the dependency $bc \to a$ contains the extraneous attribute c in the left-hand side, because changing $bc \to a$ by $b \to a$, we have that Σ^+ is the same.*

The equivalent case, in the right-hand side, is right-reduction. An attribute y is extraneous in the right-hand side of a dependency $\sigma \in \Sigma$ if removing y from the right-hand side in σ does not change Σ^+.

Definition 3.2. *A set of dependencies Σ is **non redundant** if and only if $(\Sigma \setminus \sigma)^+ \neq \Sigma^+$ for all $\sigma \in \Sigma$.*

If a set of dependencies Σ is non redundant and all the right-hand sides are singletons then Σ is **right-reduced** (Definition 5.6 in [13]).

Since $\{\, X \to yz \,\} \equiv \{\, X \to y, X \to z \,\}$, there is no difference between considering a cover such that all the right-hand sides are singletons, or just *joining* in one single dependency all those dependencies with the same left-hand side. Actually, when the right-hand sides are singletons, the number of dependencies is "artificially" increased.

Definition 3.3. *A set of dependencies Σ is **direct** if and only if* $\mathrm{closure}_{\Sigma}(X)$ *can be computed with a single pass of* Σ*, for all* $X \subseteq \mathcal{U}$.

As we stated in Section 1, we assume that the computation of $\mathrm{closure}_{\Sigma}$ is performed by the algorithm **Closure** or **LinClosure**. The Definition 3.3 means that computing the closure of any set of attributes $X \subseteq \mathcal{U}$ implies only one complete exploration of Σ. Actually, this unique exploration represents a very important property, especially when Σ is large or very large. Moreover, it should also be noticed that (i) a cover Σ is direct regardless of how it is represented or sorted, and also (ii) Σ is direct for all possible sets of attributes $X \subseteq \mathcal{U}$.

Definition 3.4. *A set of dependencies Σ is **minimal** if and only if* $|\Sigma| \leq |\Sigma'|$ *for all* Σ' *such that* $\Sigma^{+} = \Sigma'^{+}$.

It is important to notice that when a cover is minimal, it is also non redundant, but the opposite does not necessarily hold. Moreover, let us recall also the importance of computing a non redundant cover given a set of dependencies Σ, and that computing a non redundant cover is equivalent to the logical implication problem.

In the next sections we review three different and major types of covers that are of interest in the RDBM, in Logic, and in FCA.

3.2. The Canonical-Direct Basis

The Canonical-Direct Basis is defined with different characterizations in [12] and aims to remove the redundancy caused by augmentation. The computation of this cover is performed in three steps:

1. All the dependencies in Σ must have one single attribute in the right-hand side, as it was already the case above for the computation of the minimal cover. Again this is performed by simply replacing a dependency $X \rightarrow Y$ by the dependencies $X \rightarrow y_i$ for all $y_i \in Y$.
2. Σ is closed by "pseudo-transitivity", that is, Augmentation plus transitivity. Then dependencies $X \rightarrow y$ such that $y \in X$ are removed.
3. Σ is left-reduced (see Definition 3.1 and step 2 in the construction of the minimal cover).

It can be noticed that there is no removal of redundant dependencies. The only source of redundancy that is taken into account and removed is the one generated by the application of augmentation, but not of transitivity. The Canonical-Direct Basis is not necessarily minimal, nor it is non-redundant, but it is direct.

Example 3.2. *We continue with this example:* $\Sigma = \{\, a \rightarrow b, b \rightarrow ac, a \rightarrow c, bc \rightarrow a \,\}$. *Applying step 1 produces* $\Sigma = \{\, a \rightarrow b, b \rightarrow a, b \rightarrow c, a \rightarrow c, bc \rightarrow a \,\}$. *Here, step 2 consists in closing* Σ *by pseudo-transitivity, which outputs:* $\Sigma' = \{\, a \rightarrow b, b \rightarrow a, b \rightarrow c, a \rightarrow c, bc \rightarrow a, ac \rightarrow a, ab \rightarrow a, ab \rightarrow b, ac \rightarrow c \,\}$. *When applying step 2 to left-reduce* Σ*, and since* $bc \rightarrow a$ *can be left-reduced to* $b \rightarrow a$*, the following equivalent set is obtained and constitutes the final result:* $\Sigma''' = \{\, a \rightarrow b, b \rightarrow a, b \rightarrow c, a \rightarrow c \,\}$*, Since there is no removal of redundant dependencies, then dependency* $a \rightarrow c$ *appears in this canonical-direct base.*

We need to notice that this base is also characterized in Formal Concept Analysis by the so called **minimal generators** (or minimal generating set). A minimal generator is defined in [9] (Section 2.3.3) as follows: *A generating set of a closed set A is a subset $S \subseteq A$ such that $A = S''$, and, obviously, a minimal generating set of A is a subset of A minimal with respect to this property.* A similar definition exists in [18]: *A minimal generator B of a closed set F is also called a base for F , i.e. $\phi(B) = F$ and $\phi(A) \subset \phi(B)$ for every $A \subset B$, or a free subset, i.e. for every $x \in B$, $x \in \phi(B \setminus x)$* (where ϕ is a closure operator). What is also relevant is that, in this same paper, the characterization of the canonical direct base Σ_{cdb} is defined as follows: $\Sigma_{cdb} = \{\, B \to \text{closure}_\Sigma(B) \setminus B : B \subseteq \mathcal{U}$ is a minimal generator of $\text{closure}_\Sigma(B) \,\}$. That is, the left-hand sides of a Canonical-Direct Basis are the set of minimal generators of the implicit closure operator.

3.3. The Minimal Cover in the RDBM and its variations

Below we introduce the **Minimal Cover**, which is very popular among the RDBM community and can be found in most of the database textbooks under different names (see Table 1). This cover aims to remove the redundancy caused by both augmentation and transitivity.

Name	Ref	Where
Canonical Cover	Maier [13]	p. 79, Section 5.6
Minimal Cover	Ullman [3]	p. 390
Minimal Cover	Abiteboul [14]	p. 286, Exercice 11.16
Irreducible Set of Dependencies	Date [19]	p. 341, Section 11.6
Minimal Cover	Elmasri [20]	p. 549, Section 16.1.3
Canonical Cover	Silberschatz [21]	p. 324, Section 7.4.3

Table 1
References to the Minimal Cover in RDBM textbooks.

The computation of the Minimal Cover is performed in three steps:

1. All the dependencies in Σ must have only one single attribute in the right-hand side. This is performed by simply replacing a dependency $X \to Y$ by the dependencies $X \to y_i$ for all $y_i \in Y$.
2. Σ is left-reduced. This is performed by changing a dependency $X \to y$ by a dependency $X' \to y$, where $X' \subset X$, whenever $(\Sigma \setminus \{X \to Y\} \cup \{X' \to Y\})^+ \equiv \Sigma^+$ (see Definition 3.1).
3. Redundant dependencies are removed from Σ (see Definition 3.2).

It is important to notice that the order of steps 2 and 3 is relevant and mandatory. Section 5.3 in [13] includes a discussion explaining why left-reduction needs to be performed before the removal of redundant dependencies. The output of computing the Minimal Cover depends also on the order in which dependencies are processed. As a consequence it comes that there may be different minimal covers for the same Σ. Finally, the Minimal Cover does not ensure directness.

Example 3.3 (Adapted from Section 5.2 in [13]). *Let us suppose that we have the following set of dependencies:* $\Sigma = \{\, a \to b, b \to ac, a \to c, bc \to a \,\}$*. Applying step 1 outputs* $\Sigma = \{\, a \to b, b \to a, b \to c, a \to c, bc \to a \,\}$*. Then step 2 is applied to left-reduce* Σ*. Since* $bc \to a$ *can be left-reduced to* $b \to a$ *(thanks to Augmentation), then the following equivalent set is produced:* $\Sigma' = \{\, a \to b, b \to a, b \to c, a \to c \,\}$*. Finally, applying step 3 outputs:* $\Sigma'' = \{\, a \to bc, b \to a \,\}$*. By contrast, let us assume that the order of* Σ *is changed as follows:* $\Sigma = \{\, a \to b, a \to c, b \to ac, bc \to a \,\}$*. Applying step 1 yields the same set as above:* $\Sigma = \{\, a \to b, a \to c, b \to a, b \to c, bc \to a \,\}$*. When applying step 2, it comes:* $\Sigma' = \{\, a \to b, a \to c, b \to a, b \to c \,\}$*. And finally applying step 3 outputs the base* $\Sigma' = \{\, a \to b, b \to ac \,\}$*.*

3.4. The Duquenne-Guigues basis

The **Duquenne-Guigues basis** [17, 8], also called the *Canonical Basis* in the FCA community, is the cover based on pseudo-closed sets [8]. More precisely, it is defined as follows:

Definition 3.5. *The* ***Duquenne-Guigues basis*** *of a set of dependencies* Σ *is defined as*

$$\{\, X \to \mathrm{closure}_\Sigma(X) \mid X \subseteq \mathcal{U} \text{ and } X \text{ pseudoclosed} \,\}$$

where the definition of a pseudo-closed set of attributes w.r.t. a set of dependencies Σ is:

Definition 3.6. *Let* Σ *be a set of dependencies, and* $\mathcal{U}$ *the related set of attributes.* $X \subseteq \mathcal{U}$ *is* ***pseudoclosed*** *if:*

1. $X \neq \mathrm{closure}_\Sigma(X)$*, that is,* X *is not closed.*
2. *If* $Y \subset X$ *is a proper subset of* X *and pseudo-closed, then* $\mathrm{closure}_\Sigma(Y) \subseteq X$*.*

This base is not direct, but it is minimal and non-redundant. According to [12], this base is also presented by Maier in [13], where it is called the *Minimum Cover*: *"It has been obtained independently (and with different formulations) by Maier (Minimum Cover), and Guigues and Duquenne (Duquenne-Guigues basis)"*. We interpret the expression *with different formulations* as the fact that the left and right-hand sides of the Duquenne-Guigues basis can be further left/right-reduced. This is: modulo left/right-reduction, the Minimum Cover and the Duquenne-Guigues basis are essentially the same. We note that [13] (Section 5.6.2) presents a way to compute a Minimum Cover out of a non redundant set of dependencies. However, at present, the use and popularity of the Duquenne-Guigues basis seems to be rather restricted within the FCA community [9, 8].

Example 3.4. *Let us consider Example 3.1:* $\Sigma = \{\, a \to b, b \to ac, a \to c, bc \to a \,\}$*.*

As in Σ *the pseudo-closed sets are simply* $\{\, a \,\}$ *and* $\{\, b \,\}$*, the following Duquenne-Guigues basis is obtained:* $\Sigma' = \{\, a \to bc, b \to ac \,\}$*.*

3.5. Some Notes on Complexity

Concerning the complexity of the computation of the different bases, in all cases the worst-case scenario is that of a base of exponential size with respect to the number of attributes as well as the computing time of such bases.

In [22] it is proved that the complexity of computing the Canonical-Direct Basis is of order $\mathcal{O}(n^2\ (\frac{m}{2})^2\ 2^m))$, where m is the number of attributes and n is the number of records (the size of the dataset). In [16] it is proved that any algorithm computing the dependencies that hold in a dataset has a complexity of $\mathcal{O}(\binom{n}{\frac{n}{2}})$, where n is the number of attributes. As for the Duquenne-Guigues basis, in [23] is is proved that (a) the size of a Duquenne-Guigues basis is exponential in the worst-case and that (b) to estimate its size without computing it is #P-hard. In fact, deciding whether a set is a pseudo-closed set is a coNP-complete problem [24].

4. Discussion and Conclusion

Relating the three main covers plus the D-Basis is relevant and very interesting. Actually, while the D-Basis and the Canonical-Direct Basis are related by a subset relationship, such a relationship is not known to exist between the Duquenne-Guigues basis and the Minimum Cover, or between the Canonical-Direct Basis and the Duquenne-Guigues basis. In fact, in [12], in the last sentence before the acknowledgements page 28, it is stated that:

> *We conclude that this paper is contradicting a conjecture of the literature (in [37]). Indeed, one observes that the premise of implication (10) of Σ_{cd} (a direct cover) is not contained in a premise of any implication of Σ_{can} (the Duquenne-Guigues base).*

This sentence goes back to a conjecture stated in [25] and can be interpreted as follows. The left-hand sides of a Duquenne-Guigues basis, which is minimal, may not be a subset of the left-hand sides of a direct cover.

The RDBM, Logic, and FCA fields are addressing two different, yet related problems: the implication problem and the computation of a compact and representative set –cover or base– of a complete set of dependencies. Although the first question is solved in the three fields thanks to the same algorithms, that is, Closure or Linclosure, this unanimity disappears in confronting with the choice of a cover. Table 2 summarizes the three types of covers reviewed in Section 3, plus the D-Basis. It can be noticed that the Canonical-Direct Basis and the D-Basis do not keep dependencies that can be inferred by the application of the augmentation axiom: $X \rightarrow Y \models XZ \rightarrow YZ$, while they include dependencies that can be inferred by transitivity. This additional amount of information is enough to make direct these two covers. By contrast, the Minimal Cover does not contain dependencies that can be inferred by augmentation or transitivity as they are removed in the last step of the computation. And this explains why the Minimal Cover is not direct.

We have there a kind of "no free lunch theorem": the more information a base is keeping the more direct the base can be. By contrast, minimality and non redundancy do not favor directness.

	Canonical Minimal	Canonical Direct	Duquenne-Guigues Minimum	D-Basis
(found in)	RDBM	RDBM, Logic	FCA/RDBM	Lattice Th.
Minimum	no	no	yes	no
Direct	no	yes	no	yes
Redundant	no	yes	no	yes
Unique	no	yes	yes	yes

Table 2
Comparing the characteristics of the four bases.

The lack of unanimity can also be noticed in the RDBM only. Indeed textbooks such as [14, 13, 3, 19, 20, 21] tend to present the Minimal Cover as the preferred cover, while algorithms computing FDs output the Canonical-Direct Basis [26]. This can be interpreted as follows: textbooks are preferring a cover left-reduced and non-redundant, and, hence, containing less and more compact information. However, for discovering FDs, a left-reduced cover is computed without removing redundant dependencies. A possible explanation is that algorithms computing FDs take a dataset as input. Then it is easy to perform a left-reduction w.r.t. the input dataset by removing an attribute from the left-hand side and test whether the dependency still holds. However, a redundancy test is different in the sense that it can only be performed w.r.t. a set of dependencies, *once this set has completely been computed.* Such a test is of a different nature as it is not performed w.r.t. a dataset. Moreover, this does not prevent any algorithm from performing it.

Although the Minimum Cover (or Duquenne-Guigues basis) is also introduced in the RDBM field, it has not enjoyed the same popularity as the Minimal Cover. In fact, the Minimum Cover is introduced and discussed only in Maier [13]. This lack of popularity is probably due to the rather intricate characterization of the Minimum Cover and the related algorithm at that time (see for example Section 5.6, Chapter 5 in [13]). This is especially true when compared with the simplicity and expressiveness of the presentation of the Minimal Cover. The characterization of Minimum Cover cannot compete either with the clear characterization of the Duquenne-Guigues basis in FCA, or the simplicity of the NextClosure Algorithm in [9]. However, this "simplicity" is not for free and it comes at the expense of the whole theoretical framework of FCA.

The choice of a cover can be decided depending on the performance that Closure or Linclosure are offering. Both Closure and Linclosure are closely dependent on the "nature" of the set of dependencies Σ. By "nature" we mean the different characteristics explained in Section 3, which have an impact on the amount of information as well as on the number of dependencies considered. If Σ is a direct cover (Definition 3.3), both algorithms have to perform only a single pass of their outer loop. If the cover is not direct, e.g., Canonical-Direct Basis or Duquenne-Guigues basis, then, the number of iterations of the outer loop may be larger, while, at the same time, the number of iterations of the inner loop may be shorter. Again, we are facing the well-known trade-off between "expressivity and complexity": the more expressive in terms of information containment a cover is, the higher the cost of Closure and Linclosure is.

The question of the preference between the Duquenne-Guigues basis and Minimum Cover remains, even if things have changed. What is left for future work is in concern with the practical behavior of the main covers, which has to be compared and investigated from the experimental point of view, especially having in mind the needs in the RDBM, in Logic, and in FCA, but also in knowledge representation and ontology engineering. In particular, attribute exploration [9] may be a good support for evaluating the potential of the main covers studied above in ontology engineering and database design.

Acknowledgments

J. Baixeries is supported by a recognition 2021SGR-Cat (01266 LQMC) from AGAUR (Generalitat de Catalunya) and is funded by a grant AGRUPS-2022 from Universitat Politècnica de Catalunya. Mehdi Kaytoue and Amedeo Napoli are carrying out this research work as part of the French ANR-21-CE23-0023 SmartFCA Research Project.

References

[1] P. C. Kanellakis, Chapter 17 - Elements of Relational Database Theory, in: J. Van Leeuwen (Ed.), Formal Models and Semantics, Handbook of Theoretical Computer Science, Elsevier, Amsterdam, 1990, pp. 1073–1156.

[2] R. Fagin, M. Y. Vardi, The Theory of Data Dependencies - An Overview, in: Proceedings of the 11th International Colloquium on Automata, Languages, and Programming, Lecture Notes in Computer Science 172, Springer, 1984, pp. 1–22.

[3] J. D. Ullman, Principles of Database and Knowledge-Base Systems, Volume I, volume 14 of *Principles of computer science series*, Computer Science Press, 1988.

[4] A. Horn, On sentences which are true of direct unions of algebras, J. Symb. Log. 16 (1951) 14–21.

[5] R. A. Kowalski, Logic for problem solving, volume 7 of *The computer science library : Artificial intelligence series*, North-Holland, 1979.

[6] P. Padawitz, Computing in Horn Clause Theories, volume 16 of *EATCS Monographs on Theoretical Computer Science*, Springer, 1988.

[7] Y. Crama, P. L. Hammer, Boolean Functions - Theory, Algorithms, and Applications, volume 142 of *Encyclopedia of mathematics and its applications*, Cambridge University Press, 2011.

[8] B. Ganter, R. Wille, Formal Concept Analysis - Mathematical Foundations, Springer, 1999.

[9] B. Ganter, S. A. Obiedkov, Conceptual Exploration, Springer, 2016.

[10] Y. Sagiv, C. Delobel, D. S. P. Jr., R. Fagin, An Equivalence Between Relational Database Dependencies and a Fragment of Propositional Logic, J. ACM 28 (1981) 435–453.

[11] T. Ibaraki, A. Kogan, K. Makino, Functional dependencies in horn theories, Artif. Intell. 108 (1999) 1–30.

[12] K. Bertet, B. Monjardet, The multiple facets of the canonical direct unit implicational basis, Theor. Comput. Sci. 411 (2010) 2155–2166.

[13] D. Maier, The Theory of Relational Databases, Computer Science Press, 1983.

[14] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison-Wesley, 1995.

[15] C. Beeri, P. A. Bernstein, Computational problems related to the design of normal form relational schemas, ACM Trans. Database Syst. 4 (1979) 30–59.

[16] H. Mannila, K. Räihä, Design of Relational Databases, Addison-Wesley, 1992.

[17] V. Duquenne, J. Guigues, Mininal family of informative implications resulting from a binary data table, Mathematics and Humanies Sciences 24 (1986) 5–18.

[18] K. Bertet, C. Demko, J.-F. Viaud, C. Guérin, Lattices, closures systems and implication bases: A survey of structural aspects and algorithms, Theoretical Computer Science 743 (2016).

[19] C. J. Date, An introduction to database systems (7. ed.), Addison-Wesley-Longman, 2000.

[20] R. Elmasri, S. B. Navathe, Fundamentals of Database Systems, 3rd Edition, Addison-Wesley-Longman, 2000.

[21] A. Silberschatz, H. F. Korth, S. Sudarshan, Database System Concepts, Seventh Edition, McGraw-Hill Book Company, 2020.

[22] J. Liu, J. Li, C. Liu, Y. Chen, Discover dependencies from data—a review, Knowledge and Data Engineering, IEEE Transactions on 24 (2012) 251 – 264. doi:10.1109/TKDE.2010.197.

[23] S. O. Kuznetsov, On the intractability of computing the duquenne-guigues base, JUCS - Journal of Universal Computer Science 10 (2004) 927–933. URL: https://doi.org/10.3217/jucs-010-08-0927. doi:10.3217/jucs-010-08-0927. arXiv:https://doi.org/10.3217/jucs-010-08-0927.

[24] M. A. Babin, S. O. Kuznetsov, Computing premises of a minimal cover of functional dependencies is intractable, Discrete Applied Mathematics 161 (2013) 742–749. URL: https://www.sciencedirect.com/science/article/pii/S0166218X12004064. doi:https://doi.org/10.1016/j.dam.2012.10.026.

[25] K. Ben-Khalifa, S. Motameny, Horn representation of a concept lattice, Int. J. Gen. Syst. 38 (2009) 469–483.

[26] T. Papenbrock, J. Ehrlich, J. Marten, T. Neubert, J.-P. Rudolph, M. Schönberg, J. Zwiener, F. Naumann, Functional dependency discovery: An experimental evaluation of seven algorithms, Proc. VLDB Endow. 8 (2015) 1082–1093.