Discovering (frequent) Constant Conditional Functional Dependencies

Thierno Diallo, Noël Novelli, Jean-Marc Petit

Discovering (frequent) constant conditional functional dependencies

Thierno Diallo
Université de Lyon, CNRS, INSA-Lyon, LIRIS, UMR5205,
and Orchestra Networks, Paris, France.
E-mail: thierno.diallo@liris.cnrs.fr

Noël Novelli*
Université de la Méditerranée, CNRS, LIF, UMR6166, France.
E-mail: noel.novelli@lif.univ-mrs.fr

Jean-Marc Petit
Université de Lyon, CNRS, INSA-Lyon, LIRIS, UMR5205, France.
E-mail: jean-marc.petit@insa-lyon.fr

Abstract

Conditional Functional Dependencies (CFDs) have been recently introduced in the context of data cleaning. They can be seen as an unification of Functional Dependencies (FD) and Association Rules (AR) since they allow to mix attributes and attribute/values in dependencies. In this paper, we introduce our first results on constant CFD inference. Not surprisingly, data mining techniques developed for functional dependencies and association rules can be reused for constant CFD mining. We focus on two types of techniques inherited from FD inference: the first one extends the notion of agree sets and the second one extends the notion of non-redundant sets, closure and quasi-closure. We have implemented the latter technique on which experiments have been carried out showing both the feasibility and the scalability of our proposition.

1 Introduction

Initial work on dependency-based data quality methods focused on traditional dependencies, such as Functional Dependencies (FD), that were mainly developed for database design 40 years ago. Their expressiveness often limits the capture of inconsistencies or the specification of dependency rules. Even if some error measures have been defined for FDs [18], none of them can easily capture the frequency (or support) measure popularized by association rules (AR) mining [2]. These limitations highlighted the need for extending either FDs to take into account attribute/values or inversely, extending ARs to take into account attributes.

In this setting, Conditional Functional Dependencies (CFD) have been recently introduced in [5, 10] as a compromise to bridge the gap between these two notions. They can be seen as an unification of Functional Dependencies

(FD) and Association Rules (AR) since they allow to mix attributes and attribute/values in dependencies as illustrated by the following example.

Example 1.1 *We borrow the running example given in [5]. Let cust be a relation symbol describing a customer with country code (CC), area code (AC), phone number (PN), name (NM), street (STR), city (CT) and zip code (ZIP). A relation r_0 of cust is shown in Figure 1.*

Let $f_1 : CC, AC, PN \rightarrow STR, CT, ZIP$, $f_2 : CC, AC \rightarrow CT, ZIP$ and $f_3 : CC, ZIP \rightarrow STR$ be three FDs. r_0 satisfies f_1 and f_2 but violates f_3 (e.g. tuples t_4 and t_5).

In contrast CFDs are contraints that hold on a subset of tuples rather than on the entire relation. So the basic idea of CFDs is to define through a selection formula (involving only equality) a subset of a relation on which some FDs hold. For instance on $\sigma_{CC=44}(r_0)$, the FD $f_3 : CC, ZIP \rightarrow STR$ holds. Technically this contraint is denoted by the CFD $\phi_0 = (CC, ZIP \rightarrow STR, (44, \_ \parallel \_))$, where the symbol '$\parallel$' is used to separate the left-hand side from the right-hand side of the dependency and the symbol '_' represents any possible value. CFDs that holds on r_0 include also the following (and more):
$\phi_1 : (CC, AC, PN \rightarrow STR, CT, ZIP(01, 908, \_ \parallel \_, NYC, \_))$
$\phi_2 : (CC, AC, PN \rightarrow STR, CT, ZIP(01, 212, \_ \parallel \_, PHI, \_))$
$\phi_3 : (CC, AC \rightarrow CT(01, 215 \parallel PHI))$

r_0	CC	AC	PN	NM	STR	CT	ZIP
t_1 :	01	908	1111111	Mike	Tree Ave.	NYC	07974
t_2 :	01	908	1111111	Rick	Tree Ave.	NYC	07974
t_3 :	01	212	2222222	Joe	Elm Str.	NYC	01202
t_4 :	01	212	2222222	Jim	Elm Str.	NYC	01202
t_5 :	01	215	3333333	Ben	Oak Av.	PHI	01202
t_6 :	44	131	4444444	Ian	High St.	EDI	03560
t_7 :	44	140	5555555	Kim	High St.	PHI	03560

Figure 1: Relation r_0 on *cust*

From a data mining point of view, CFDs offer new opportunities to reveal data inconsistencies or new knowledge nuggets from existing tabular datasets.

Data cleaning is the main application of CFDs [5]. The first step of this application is detecting CFD violation, i.e. given a relation r on R and a set Σ of CFDs on R, find all the tuples in r that violate some CFD in Σ. In [5] authors propose `SQL` queries to check single CFD violation, and then techniques for single CFD are generalized to find violation of multiple CFDs. The second and last step is CFD repairing where they allow attribute-value modifications as a repair operation.

Clearly, it might be useful to discover CFDs from a particular database instance. In this paper, we address the following problem, called *CFD inference* :

Given a relation r, discover a cover of CFDs satisfied in r.

In [5, 10], CFDs has been proposed and studied mainly from a theoretical perspective, their underlying application being data cleaning. They revise

classical problems (implication, consistency, axiomatization...) in data dependencies for CFDs.
In [13], authors study the characterization and the generation of pattern tableaux to realize the full potential of CFDs. In [22], a hierarchy of CFDs, FDs and ARs has been proposed along with some theoretical results on pattern tableaux equivalence. They use the work of [6], based on horizontal decomposition of a relation, as a way to represent and reason on CFDs.

As far as we know, only two contributions have been made for CFD mining [7, 11] while plenty of contributions have been proposed for FD inference and AR mining (see for example [2, 25, 16, 19, 23] in the context of this paper). In [7], authors propose a tool for data quality management which suggests possible rules and identify conform and non-conform records. They present effective algorithms for discovering CFDs and dirty values in a data instance, but the CFDs discovered may contain redundant patterns. In [11], authors proposed three methods to discover CFDs. The first one is called CFDMiner which mines only constant CFDs i.e. CFDs with constant patterns only. CFDMiner is based on techniques for mining closed itemsets [25]. The two other ones, called CTANE and FastCFD, were developed for general (non constant) CFDs discovery. CTANE and FastCFD are respectively extensions of well known algorithms TANE [16] and FastFD [27] for mining FDs.

Contribution In this paper, we introduce our first results on CFD inference which can be seen as a clarification of some simple and basic notions underlying CFDs. Not surprisingly, we point out how data mining techniques developed for functional dependencies and association rules can be reused for constant CFD mining. We focus on two types of techniques: the first one extends the notion of agree sets and the second one extends the notion of non-redundant sets, closure and quasi-closure. We have implemented the latter techniques on which experiments have been carried out showing both the feasibility and the scalability of our proposition.

Paper Organization In Section 2, basic notations for CFDs are given. In Section 3, news notations are introduced to simplify CFDs. The first proposition based on conditional agree sets to discover constant CFDs is made in Section 4. The second proposition is based on the extension of Fun [23]: its main interest is to take into account the frequency of CFDs to discover frequent constant CFDs (Section 5). The experimental results are presented in Section 6 and then we conclude and give some perspectives of this work in the last Section.

2 Preliminaries

We shall use classical database notions (e.g. [1] and CFDs terminology [5, 10, 13, 22]). We do not distinguish a relation symbol form its schema, i.e. a relation symbol R is seen as a set of attributes from some universe. Each attribute A has a domain, denoted by $DOM(A)$. Given a relation r over R and $A \in R$, active domain of A in r is denoted by $ADOM(A, r)$. A relation is a set of tuples and the projection of a tuple t on an attribute set X is denoted by $t[X]$. Letters from the beginning of the alphabet $(A, B, C, \ldots)$ shall represent single attribute whereas letters from the end of the alphabet $(X, Y, Z, \ldots)$ attribute sets. For

convenience, XY will refer to as $X \cup Y$.

Consider a relation schema R, the syntax of a CFD is given as follows: A Conditional Functional Dependence (CFD) ρ on R is a pair $(X \rightarrow Y, T_p)$ where (1) $XY \subseteq R$, (2) $X \rightarrow Y$ a standard Functional Dependency (FD) and (3) T_p is a *pattern tableau* with attributes in R. For each $A \in R$ and for each pattern tuple $t_p \in T_p$, $t_p[A]$ is either a constant in $DOM(A)$, or an *'unnamed variable'* denoted by '_', or an empty variable denoted by '$*$' which indicates that the corresponding attribute does not contribute to the pattern (i.e. $A \notin XY$).

The semantics of a CFD extends the semantics of FD with mainly the notion of matching tuples.

Let r be a relation over R, $X \subseteq R$ and T_p a pattern tableau over R. A tuple $t \in r$ *matches* a tuple $t_p \in T_p$ over X, denoted by $t[X] \asymp t_p[X]$, iff for each attribute $A \in X$, either $t[A] = t_p[A]$, or $t_p[A] =$'_', or $t_p[A] =$'$*$'.

Let r be a relation over R and $\rho = (X \rightarrow Y, T)$ a CFD with $XY \subseteq R$. We say that r satisfies ρ, denoted by $r \models \rho$, iff for all t_i,$t_j \in r$ and for all $t_p \in T$, if $t_i[X] = t_j[X] \asymp t_p[X]$ then $t_i[Y] = t_j[Y] \asymp t_p[Y]$.

Example 2.1 *The relation r_0 of the Figure 1 satisfies CFDs ϕ_0, ϕ_1 and ϕ_3 given in Example 1.*

We say that r satisfies a set Σ of CFD over R, denoted by $r \models \Sigma$ if $r \models \rho$ for each CFD $\rho \in \Sigma$.

Let Σ_1 and Σ_2 be two sets of CFD defined over the same schema R. We say that Σ_1 is equivalent to Σ_2 denoted by $\Sigma_1 \equiv \Sigma_2$ iff for any relation r over R, $r \models \Sigma_1$ iff $r \models \Sigma_2$.

An FD $X \rightarrow Y$ is a special case of CFD $(X \rightarrow Y, t_p)$ where t_p is a single pattern tuple and for each $B \in XY$, $t_p[B] =$ '_ '.

Compare to FDs, note that a single tuple relation may violate a CFD. It may occur when the pattern tableau has at least one row with at least one constant on the right-hand side. Given a relation, the satisfaction of a CFD has to be checked with both every single tuple and every couple of tuples. As for classical FD, the non-satisfaction is much easier to verify: it is enough to exhibit a counter-example, i.e. either a single tuple or a couple of tuples.
More formally, we get:
r violate a CFD $\rho = (X \rightarrow Y, T)$, denoted by $r \not\models \rho$, iff

- there exists a tuple $t \in r$ and a pattern tuple $t_p \in T$ such that $t[X] \asymp t_p[X]$ and $t[Y] \not\asymp t_p[Y]$ or
- there exists $t_i, t_j \in r$ and a pattern tuple $t_p \in T$ such that $t_i[X] = t_j[X] \asymp t_p[X]$ and $t_i[Y] \neq t_j[Y]$.

As we will see later, the first condition will turn out to be useless in the context of CFD inference.

Example 2.2 *In Figure 1 the relation r_0 does not satisfy these two CFDs:*

- $\phi_2 : (CC, AC, PN \rightarrow STR, CT, ZIP(01, 212, \_ \parallel \_, PHI, \_))$
  Indeed, t_3 violates $\phi_2 : t_3\,[CC, AC, PN] \asymp (01, 212, \_)$ but $t_3\,[STR, CT, ZIP] \not\asymp (\_, PHI, \_)$.

- $\phi_4 = (CC, CT \rightarrow ZIP(01, \_ \parallel \_))$. *Indeed,* t_2 *and* t_3 *violate* ϕ_4 *since* $t_2[CC, CT] = t_3[CC, CT] \asymp (01, \_)$, *but* $t_2[ZIP] \neq t_3[ZIP]$.

Let Σ be a set of CFD and $(X \rightarrow Y, T_p)$ a single CFD over R. Σ implies $(X \rightarrow Y, T_p)$ denoted by $\Sigma \models (X \rightarrow Y, T_p)$ iff for every relation r over R, if $r \models \Sigma$ then $r \models (X \rightarrow Y, T_p)$.
$\Sigma \not\models (X \rightarrow Y, T_p)$ iff there exists a relation r over R such that $r \models \Sigma$ but $r \not\models (X \rightarrow Y, T_p)$.

A CFD $(X \rightarrow Y, T_p)$ is in the *normal form* [10], when $|Y| = 1$ and $|T_p| = 1$. So a normalized CFD has a single attribute on the right-hand side and its pattern tableau has only one single tuple.

Proposition 2.1 *[10] For any set* Σ *of CFD there exists a set* Σ_{nf} *of CFD such that each CFD* $\rho \in \Sigma_{nf}$ *is in the normal form and* $\Sigma \equiv \Sigma_{nf}$.

In the sequel we consider CFDs in their normal form, unless stated otherwise.
A CFD $(X \rightarrow A, t_p)$ is called:

- a *constant CFD* if $t_p[XA]$ consists of constants only, i.e. $t_p[A]$ is a constant and $t_p[B]$ is also a constant for all $B \in X$.
- a *variable CFD* if the right hand side of its pattern tuple is the unnamed variable '_', i.e. $t_p[A] =$ '_', the left-hand side involving either constants or '_'.

Proposition 2.2 *[10] For any set* Σ *of CFD over a schema* R, *there exists a set* Σ_c *of constant CFDs and a set* Σ_v *of variable CFDs over* R *such that* $\Sigma \equiv \Sigma_c \cup \Sigma_v$.

3 New notations for CFDs

We now introduce new notations for representing CFDs, which will turn out to be very convenient to express known results in database theory.

3.1 Search space for constant CFDs

Usually, the search space for FDs and ARs is a powerset of the set of attributes (or items). With CFDs, attributes have to be considered together with their possible values. This is defined as follows for constant CFDs.

Definition 3.1 *Let* R *be a relation symbol. The search space of constant CFDs over* R, *denoted by* $SP_{CFD}(R)$, *is defined as follows:*

$$SP_{CFD}(R) = \{(A, a) \mid A \in R, a \in DOM(A)\}$$

Formally, the search space is the powerset of $SP_{CFD}(R)$. This search space for constant CFD is infinite if at least one of the attributes of R has an infinite domain.
Let $\rho = (A_1 \dots A_n \rightarrow A, t_p[A_1 \dots A_n A])$ be a constant CFD over R.
Note that ρ can be seen as syntactically equivalent to $\overline{X} \rightarrow \overline{A}$, with $\overline{X} = \{(A_1, t_p[A_1]), \dots, (A_n, t_p[A_n])\} \subseteq SP_{CFD}(R)$ and $\overline{A} = (A, t_p[A]) \in SP_{CFD}(R)$.

In the sequel, constant CFDs will be often represented using this new notation, with elements of $SP_{CFD}(R)$ only. Given $\overline{A} = (A, v) \in SP_{CFD}(R)$, we note $\overline{A}.att$ and $\overline{A}.val$ the values A and v respectively. By extension, given $\overline{X} \subseteq SP_{CFD}(R)$, $\overline{X}.att$ represents the union of attributes belonging to $\overline{X}$, i.e. $\overline{X}.att = \bigcup_{\overline{A} \in \overline{X}} \overline{A}.att$

The search space of constant CFDs is now defined to take into account a relation r over R.

Definition 3.2 *Let R be a relation symbol and r a relation over R. The search space of constant CFDs for r, denoted by $ASP_{CFD}(R, r)$, is defined as:*

$$ASP_{CFD}(R, r) = \{(A, a) \mid A \in R, a \in ADOM(A, r)\}$$

Since we consider finite relation only, this set is finite.

Example 3.1 *Let r be the relation in Figure 2. For sake of clearness we denote the couple (A_i, v) by $A_i v$. We have:*
$ASP_{CFD}(ABCD, r) = \{A0, A2, B0, B1, B2, C0, C3, D1, D2\}$

r	A	B	C	D
t_1 :	0	1	0	2
t_2 :	0	1	3	2
t_3 :	0	0	0	1
t_4 :	2	2	0	1
t_5 :	2	1	0	1

Figure 2: A relation r over $R = \{A, B, C, D\}$

3.2 Minimality and covers of constant CFDs

A CFD $\overline{X} \rightarrow \overline{A}$ over R is said to be *trivial* if $\overline{A}.att \in \overline{X}.att$. In the sequel we consider nontrivial CFDs only.

A constant CFD $\overline{X} \rightarrow \overline{A}$ is said to be *left-reduced* on r if for any $\overline{Y}.att \subset \overline{X}.att$, $r \not\models \overline{Y} \rightarrow \overline{A}$.

Intuitively none of its left-hand side attributes can be removed and none of the constants in its left-hand side pattern can be "upgraded" to '_'.

A *minimal CFD* [10] ρ on r is a non trivial, left-reduced CFD such that $r \models \rho$.

A *canonical cover* of a set Σ_r of CFDs is a set Σ_{cc} of minimal CFDs such that $\Sigma_r \equiv \Sigma_{cc}$.

From Propositions 2.1 and 2.2, we can now derive a new proposition whenever the set of CFDs comes from a relation.

Proposition 3.1 *For any relation r, there exists a set Σ_c of constant CFDs such that $\Sigma_r \equiv \Sigma_c$, Σ_r being the set of satisfied CFDs in r.*

The problem statement already given can be refined as follows:

> *Given a relation r, discover a canonical cover of constant CFDs satisfied in r*

Now, we have the following property related to the monotonicity of CFDs *w.r.t.* to their partial order.

Property 3.1 *Let r be a relation over $R, \overline{X}, \overline{Y} \subseteq ASP_{CFD}(R, r)$ such that $\overline{X} \subseteq \overline{Y}$ and $\overline{A} \in ASP_{CFD}(R, r)$. We have:*
$r \models \overline{X} \rightarrow \overline{A} \Rightarrow r \models \overline{Y} \rightarrow \overline{A}$ *(or equivalently* $r \not\models \overline{Y} \rightarrow \overline{A} \Rightarrow r \not\models \overline{X} \rightarrow \overline{A}$*)*

3.3 Closure operator for constant CFDs

With these new notations, closure of attribute sets and pattern tuples defined in [10] can be rewritten easily and most well known results in data dependency theory can be rephrased (mainly from FDs).

Definition 3.3 *Let Σ be a set of constant CFDs and $\overline{X} \subseteq SP_{CFD}(R)$.*
The closure of $\overline{X}$ with respect to Σ, denoted by $\overline{X}^*_\Sigma$, is defined as follows:
$\overline{X}^*_\Sigma = \{\overline{A} \in SP_{CFD}(R) \mid \Sigma \models \overline{X} \rightarrow \overline{A}\}$.

The operator $.^*_\Sigma$ defined on the powerset of $SP_{CFD}(R)$ is a closure operator, i.e. extensive, monotonic and idempotent.

Example 3.2 *Let $\Sigma = \{\phi_1, \phi_2\}$ with $\phi_1 = (A \rightarrow B, (0 \parallel 1))$ and $\phi_2 = (B \rightarrow C, (1 \parallel 2))$. For instance, we have:*
$\{(A,0)\}^*_\Sigma = \{(A,0),(B,1),(C,2)\}$, $\{(A,0),(B,2)\}^*_\Sigma = \{(A,0),(B,1),(C,2),(B,2)\}$ *and* $\{(A,1),(B,2)\}^*_\Sigma = \{(A,1),(B,2)\}$.

Property 3.2 *Let Σ be a set of constant CFDs, $\overline{X} \subseteq SP_{CFD}(R)$ and $\overline{A} \in SP_{CFD}(R)$.*
$\overline{A} \in \overline{X}^*_\Sigma$ *iff* $\Sigma \models \overline{X} \rightarrow \overline{A}$.

Definition 3.4 *Let Σ be a set of constant CFDs over R. The closed sets of Σ with respect to R, denoted by $CL(\Sigma)$, are defined as follows:*
$CL(\Sigma) = \{\overline{X} \subseteq SP_{CFD}(R) \mid \overline{X} = \overline{X}^*_\Sigma\}$.

With the new notations introduced so far, it is worth noting that the main notions useful for CFDs appear to be very close to their counterparts for functional dependencies.

4 Discovering constant CFD using conditional agree sets

We now introduce a new set called *conditional agree set*, an extension of traditional agree set [3].

We first define the conditional agree set between a single tuple and a pattern. Let r be a relation over R, $t \in r$ a single tuple, t_p a pattern tuple over R.

Definition 4.1 *A conditional agree set between a single tuple t and a pattern t_p, denoted by $ag(t, t_p)$, is defined by: $ag(t, t_p) = \{(A, t[A]) \mid t[A] \asymp t_p[A], A \in R\}$.*

Example 4.1 *Let r be the relation over $R = ABCD$ (cf. Figure 2).*
$t_p = (0, 1, 3, \_)$ *a pattern tuple.*
$ag(t_1, t_p) = \{A0, B1, D2\}$.
$ag(t_2, t_p) = \{A0, B1, C3, D2\}$.

Second we introduce the conditional agree set between two tuples and a pattern. Let r be a relation over R, t_1, t_2 two tuples of r and t_p a pattern tuple over R.

Definition 4.2 *A conditional agree set between two tuples t_1, t_2 and t_p, denoted by $ag(t_1, t_2, t_p)$, is defined by:* $ag(t_1, t_2, t_p) = \{(A, t_1[A]) \mid t_1[A] = t_2[A] \asymp t_p[A], A \in R\}$.

Example 4.2 *Let r be the relation over $R = ABCD$ (cf. Figure 2).*
$t_p = (0, 1, 3, \_)$ *a pattern tuple.*
$ag(t_1, t_2, t_p) = \{A0, B1, D2\}$.

As expected, the main information we have from $ag(t_1, t_2, t_p)$ is a counter-example when $A = 0$, $B = 1$ and $D = 2$. For instance, $\{t_1, t_2\} \not\models A0, B1, D2 \rightarrow C0$ or $\{t_1, t_2\} \not\models A0, B1, D2 \rightarrow C3$.

Now we define the conditional agree set between a relation and a pattern. Let r be a relation over R and t_p a pattern tuple over R.

Definition 4.3 *The conditional agree set between r and t_p, denoted by $ag(r, t_p)$ is defined by:* $ag(r, t_p) = \{ag(t_i, t_j, t_p) \mid t_i, t_j \in r,\ t_i \neq t_j\}$.

Lastly we define the *conditional agree set* with all possible pattern tuples of a given relation r denoted by $cag(r)$.

Definition 4.4 $cag(r) = \bigcup_{t_p} ag(r, t_p)$ *for all pattern tuples t_p.*

Clearly we have the following result.

Property 4.1 *Let t_p be the pattern tuple such that $t_p[A] =$ '_' for all $A \in R$. We have:*

$$cag(r) = ag(r, t_p)$$

Example 4.3 *Continuing the Example 4.2, we get:*
$cag(r) = \{\{A0B1D2\}, \{A0C0\}, \{C0\}, \{B1C0\}, \{A0\}, \{B1\}, \{C0D1\}, \{A2C0D1\}\}$

We are interested in enumerating the left-hand sides of all minimal CFDs satisfied in r whose right-hand side is reduced to a single couple (attribute, value).
We need to define the so-called *valid portion* of the relation r for which it makes sense to determine CFDs with such a given right-hand side, say $\overline{A}$ in $ASP_{CFD}(R, r)$. Clearly, the tuples t of r such that $t[\overline{A}.att] \neq \overline{A}.val$ are useless as well as the attribute $\overline{A}.att$.

In the sequel, we shall use the following notation:
Let $\overline{A}$ in $ASP_{CFD}(R, r)$. The *valid portion* of r with respect to $\overline{A}$, denoted by $r'(\overline{A})$, is defined as: $r'(\overline{A}) = \pi_{R-\overline{A}.att}(\sigma_{\overline{A}.att=\overline{A}.val}(r))$.

Definition 4.5 *The left-hand side of left-reduced constant CFD for $\overline{A}$, denoted by $lhs(\overline{A}, r)$, is defined by:*

$$lhs(\overline{A}, r) = \{\overline{X} \subseteq ASP_{CFD}(R - \overline{A}.att, r'(\overline{A})) \mid r \models \overline{X} \rightarrow \overline{A} \text{ and for all } \overline{Y} \subset \overline{X}, r \not\models \overline{Y} \rightarrow \overline{A}\}$$

To characterize $lhs(\overline{A}, r)$, we borrow the same principles used for FD inference [20, 9].

We first define the maximal sets (with respect to set inclusion) of attribute/values that do not satisfy the CFD.

Definition 4.6 *The maximal sets of not satisfied constant CFDs for $\overline{A}$ in r, denoted by $max(\overline{A}, r)$, is defined by:*

$$max(\overline{A}, r) = max_{\subseteq}\{\overline{X} \subseteq ASP_{CFD}(R - \overline{A}.att, r'(\overline{A})) \mid r \not\models \overline{X} \rightarrow \overline{A}\}$$

From Property 3.1, we know that maximal sets are enough to capture invalid CFDs.

Now we can bridge the gap between conditional agree sets and maximal sets. Intuitively, we need to consider elements $\overline{X}$ of $cag(r)$ such that $\overline{A}$ does not belong to $\overline{X}$ (not a counter-example).

Property 4.2 $max(\overline{A}, r) = max_{\subseteq}\{\overline{X} \in cag(r) \mid \overline{A} \notin \overline{X}\}$

Example 4.4 *Continuing the Example 4.3, we have:*
$max(A0, r) = \{\{B1C0\}, \{C0D1\}\}$
$max(A2, r) = \{\{B1C0\}, \{C0D1\}\}$
$max(B0, r) = \{\{A0C0\}, \{C0D1\}\}$
$max(B1, r) = \{\{A0C0\}, \{A2C0D1\}\}$
$max(B2, r) = \{\{A2C0D1\}\}$
$max(C0, r) = \{\{A0B1D2\}\}$
$max(C3, r) = \{\{A0B1D2\}\}$
$max(D1, r) = \{\{A0C0\}, \{B1C0\}\}$
$max(D2, r) = \{\{A0C0\}, \{B1C0\}\}$

From the *maximal sets* of elements (that do not satisfy the CFD), we have to identify the *minimal sets* of elements that satisfy the CFD.

This kind of relationship has been heavily studied in many pattern enumeration problems and has been formalized in [21].

So, to come up with the main result, we define the complement of elements of $max(\overline{A}, r)$ in $ASP_{CFD}(R - \overline{A}.att, r'(\overline{A}))$. For a given $\overline{A}$, the search space is the set of all possible couples of the form (attribute, value) from $r'(\overline{A})$.

Definition 4.7 $cmax(\overline{A}, r) = \{ASP_{CFD}(R - \overline{A}.att, r'(\overline{A})) - \overline{X} \mid \overline{X} \in max(\overline{A}, r)\}$.

Example 4.5 *Continuing the previous Example 4.4, let us consider $cmax(A0, r)$. For $A0$, we have $ASP_{CFD}(R - \overline{A}.att, r'(A0)) = \{B0, C3, D2, B1, D1\}$. So the complement of $max(A0, r)$ with respect to $ASP_{CFD}(R - \overline{A}.att, r'(A0))$ is:*
$cmax(A0, r) = \{\{B0C3D1D2\}, \{B0B1C3D2\}\}$

For the other (attribute, value) couples, we have:
$cmax(A2, r) = \{\{B2D1\}, \{B1B2\}\}$
$cmax(B0, r) = \{\{D1\}, \{A0\}\}$
$cmax(B1, r) = \{\{A2C3D1D2\}, \{A0C3D2\}\}$
$cmax(B2, r) = \{\}$
$cmax(C0, r) = \{\{A2B0B2D1\}\}$
$cmax(C3, r) = \{\}$

$cmax(D1, r) = \{\{A2B0B1B2\}, \{A0A2B0B2\}\}$
$cmax(D2, r) = \{\{B1C3\}, \{A0C3\}\}$

Now, the main result can be given. We reused the well-known connection between positive and negative borders of interesting pattern enumeration problems representable as sets [21]. This connection relies on minimal transversal of a hypergraph. A collection $\mathcal{H}$ of subsets of a finite set is a *simple hypergraph* if $\forall X \in \mathcal{H}, X \neq \emptyset$ and $(X, Y \in \mathcal{H}$ and $X \subseteq Y \rightarrow X = Y)$ [4]. A *transversal* T of $\mathcal{H}$ is a subset of R intersecting all the edges of $\mathcal{H}$, i.e. $T \cap E \neq \emptyset, \forall E \in \mathcal{H}$. A *minimal transversal* of $\mathcal{H}$ is a transversal T such that it does not exist a transversal T' of H, $T' \subset T$. The collection of minimal transversals of $\mathcal{H}$ is denoted by $\mathrm{Tr}(\mathcal{H})$.

Theorem 4.1 *Let r be a relation over R and $\overline{A} \in ASP_{CFD}(R, r)$.*

$$lhs(\overline{A}, r) = TrMin(cmax(\overline{A}, r))$$

where $TrMin(H)$ is the set of minimal transversal of the hypergraph H [21].

The proof follows from the following arguments: the search space is a finite subset lattice and the predicate is monotonic *w.r.t.* set inclusion (cf. Property 3.1). The details are omitted.

Example 4.6 *From previous examples, we have the following left-hand sides:*

lhs
$lhs(A0, r) = \{\{B0\}, \{C3\}, \{D2\}, \{B1D1\}\}$
$lhs(A2, r) = \{\{B2\}, \{B1D1\}\}$
$lhs(B0, r) = \{\{A0D1\}\}$
$lhs(B1, r) = \{\{C3\}, \{D2\}, \{A0A2\}, \{A0D1\}\}$
$lhs(B2, r) = \{\}$
$lhs(C0, r) = \{\{A2\}, \{B0\}, \{B2\}, \{D1\}\}$
$lhs(C3, r) = \{\}$
$lhs(D1, r) = \{\{A2\}, \{B0\}, \{B2\}, \{A0B1\}\}$
$lhs(D2, r) = \{\{C3\}, \{A0B1\}\}$

Let r be a relation over R. The canonical cover of satisfied CFDs in r, denoted by $\Sigma_{cc}(R, r)$, is defined by:

$$\Sigma_{cc}(R, r) = \bigcup_{\overline{A} \in ASP_{CFD}(R,r)} lhs(\overline{A}, r) \rightarrow \overline{A}$$

Example 4.7 *From the previous example, we obtain:*
$\Sigma_{cc}(R, r) = \{\ B0 \rightarrow A0;\ C3 \rightarrow A0;\ D2 \rightarrow A0;\ B1D1 \rightarrow A0;\ B2 \rightarrow A2;\ B1D1 \rightarrow A2;\ A0D1 \rightarrow B0;\ C3 \rightarrow B1;\ D2 \rightarrow B1;\ A0A2 \rightarrow B1;\ A0D1 \rightarrow B1;\ A2 \rightarrow C0;\ B0 \rightarrow C0;\ B2 \rightarrow C0;\ D1 \rightarrow C0;\ A2 \rightarrow D1;\ B0 \rightarrow D1;\ B2 \rightarrow D1;\ A0B1 \rightarrow D1;\ C3 \rightarrow D2;\ A0B1 \rightarrow D2\ \}$.

Note that the CFD $A0A2 \rightarrow B1$ does not make sense and have to be pruned in a post-precessing stage. Such kind of CFDs can be seen as a side effect of the dualization (transversal minimal computation).

As we have shown, inferring constant CFDs is an instance of the class of interesting pattern enumeration problem. Therefore, existing implementations can be easily modified to get CFDs, for instance using the `iZi` library [12].

Example 4.8 *From the previous example, let us consider the CFD* $A0, D1 \rightarrow B1$. *Interestingly, there is no tuple that matches* $(0, 1, \_, 1)$ *in the relation of Figure 2.*

The previous example points out that the method proposed in this section may produce "useless" CFDs, i.e. CFDs that do not match any tuple of r. This could be addressed by taking into account the `frequency of the CFD`, i.e. the number of tuples that match a given CFD. As a post-treatment, it requires a full scan of the database and can be easily performed.

Nevertheless, it turns out that the notion of frequency (or support which tries to capture the strength of a dependency) cannot be taken into account easily a priori: the dualization (minimal transversal computation) does not allow to take care of frequency during its computation. This is a non trivial issue which is out of the scope of this paper.

The next section proposes a new approach able to integrate the frequency constraint.

5 Frequent constant CFD discovery

We would like to be able to discover frequent constant CFDs in a relation, i.e CFDs for which the number of tuples in the relation satisfying them is above a given threshold.

Intuitively, the frequency of a CFD in a relation is the number of tuples that match its pattern tuple, i.e. the size of the corresponding selection query.

Definition 5.1 *Let* $\theta = (\overline{X} \rightarrow \overline{Y})$ *be a constant CFD over* R *and* r *a relation over* R. *The frequency of* θ *in* r, *denoted by* $freq(\theta, r)$, *is defined as follows:*

$$freq(\theta, r) = |\sigma_{\wedge_{(A,v)\in\overline{X}\cup\overline{Y}}(A=v)}(r)|$$

Let ϵ *be an integer threshold value. A CFD* θ *is said to be frequent in* r, *if* $freq(\theta, r) \geq \epsilon$.

As expected, the frequency is a monotonic predicate.

Property 5.1 *Let* r *be a relation over* R *and* $\overline{X}, \overline{Y} \subseteq ASP_{CFD}(R, r)$ *such that* $\overline{X} \subseteq \overline{Y}$ *and* ϵ *a threshold. We have:*
$freq(\overline{Y}, r) \geq \epsilon \Rightarrow freq(\overline{X}, r) \geq \epsilon$ *(or* $freq(\overline{X}, r) < \epsilon \Rightarrow freq(\overline{Y}, r) < \epsilon$*)*

We need to define a test to decide whether or not a given CFD holds in a relation. We know for a while that such a property exists for testing the satisfaction of an FD in a relation. The following property is often used: $r \models X \rightarrow Y$ iff $|\pi_X(r)| = |\pi_{XY}(r)|$.

For CFD, a similar property holds. It is stated below using the relational algebra selection operator.

Property 5.2 *Let* R *is a relation symbol,* r *is a relation over* R, $\overline{X}, \overline{Y} \subseteq ASP_{CFD}(R, r)$ *and* $C_{\overline{X}}, C_{\overline{Y}}$ *are two selection formulas over* $\overline{X}$ *and* $\overline{Y}$ *respectively.*

$r \models \overline{X} \rightarrow \overline{Y}$ *iff* $|\sigma_{C_{\overline{X}}}(r)| = |\sigma_{C_{\overline{X}} \wedge C_{\overline{Y}}}(r)|$ *where* $C_{\overline{X}} = \wedge_{(A,v)\in\overline{X}}(A = v)$ *and* $C_{\overline{Y}} = \wedge_{(A,v)\in\overline{Y}}(A = v)$

Definition 5.2 *Conditional non-redundant sets*
Let $\overline{X} \subseteq ASP_{CFD}(R,r)$ *is a set of conditional attributes.*
$\overline{X}$ *is a conditional non-redundant set in* r *if and only if* $\nexists \overline{X'} \subseteq \overline{X}$ *such that* $|\sigma_{C_{\overline{X'}}}(r)| = |\sigma_{C_{\overline{X}}}(r)|$.

The set of all conditional non-redundant sets in r is denoted by $\mathcal{NRS}_r$. Any set of conditional attributes not included in $\mathcal{NRS}_r$ is called a *conditional redundant set*.

Property 5.3 *Let* r *be a relation over* R *and* $\overline{X}, \overline{Y} \subseteq ASP_{CFD}(R,r)$ *such that* $\overline{X} \preceq \overline{Y}$. *We have:*
$\overline{Y} \in \mathcal{NRS}_r \Rightarrow \overline{X} \in \mathcal{NRS}_r$ *(or equivalently* $\overline{X} \notin \mathcal{NRS}_r \Rightarrow \overline{Y} \notin \mathcal{NRS}_r$*)*

Clearly, Apriori-like algorithms can be used to discover frequent non-redundant sets (conjunction of two monotonic predicates).

From the non-redundant sets, we extend the results given in [23, 24] (for FD inference) to propose a new characterization of the canonical cover of CFDs in which we integrate a frequency threshold. It is based on non-redundant sets, frequency, closure and quasi-closure of CFDs. The definitions follow.

Now we can redefine the closure operator (cf. Definition 3.3) in this context.

Definition 5.3 *Conditional attribute set closure in a relation*
Let $\overline{X}$ *be a set of conditional attributes,* $\overline{X} \subseteq ASP_{CFD}(R,r)$. *Its closure in* r *is defined as follows:*

$$\overline{X}^*_{\Sigma_r} = \overline{X} \cup \{\overline{A}/\overline{A}.att \in R - \overline{X}.att \wedge |\sigma_{C_{\overline{X}}}(r)| = |\sigma_{C_{\overline{X}} \wedge C_{\overline{A}}}(r)|\}.$$

We introduce the concept of quasi-closure which allows to accumulate the knowledge extracted from the subsets of the considered conditional attribute set.

Definition 5.4 *Conditional attribute set quasi-closure in a relation*
The quasi-closure of a conditional attribute set $\overline{X}$ *in* $ASP_{CFD}(R,r)$*, denoted by* $\overline{X}^{\diamond}_{\Sigma_r}$*, is defined by:*

$$\overline{X}^{\diamond}_{\Sigma_r} = \overline{X} \cup \bigcup_{\overline{A} \in \overline{X}} (\overline{X} - \overline{A})^*_{\Sigma_r}$$

According to the monotony property of the closure operator, we have: $\overline{X} \subseteq \overline{X}^{\diamond}_{\Sigma_r} \subseteq \overline{X}^*_{\Sigma_r}$.

Through the following theorem, we prove that the set of constant CFDs characterized by using the introduced concepts is the canonical cover of constant CFDs for the relation r.

Theorem 5.1

$$\Sigma_{cc_\epsilon}(R,r) = \{\overline{X} \rightarrow \overline{A} \ / \ \overline{X} \in \mathcal{NRS}_r, freq(\overline{X},r) \geq \epsilon \text{ and } \overline{A} \in \overline{X}^*_{\Sigma_r} - \overline{X}^{\diamond}_{\Sigma_r}\}$$

We omit the proof.

The theoretical framework proposed is well adapted to implement a level-wise approach for discovering CFDs from a relation. Our algorithm, called CFun, is based on the concepts of Apriori to find all conditional non-redundant sets. Once the conditional non-redundant sets discovered for each level and the corresponding frequency (count), quasi-closure and closure, discovering CFDs is trivial following Theorem 5.1. This philosophy is the same as that used for the FD inference approach called Fun [23, 24]. The pruning rule is provided by the Proposition 5.3 to extract only non-redundant sets.

Each level contains a collection of quadruplets $< \overline{X}, |\overline{X}|, \overline{X}^{\diamond}_{\Sigma}, \overline{X}^{*}_{\Sigma} >$ that respectively represents the candidate, its frequency, quasi-closure, and closure as shown in the Figure 3. The algorithm starts by initializing the first two levels 0 and 1 then it follows by a loop through levels (line 3-8). Each loop computes the closure (line 4) of non-redundant sets left in the previous level then the quasi-closure (line 5) of candidates in the current level (cf. Definition 5.3). The CFDs hold are displayed (line 6) according to the Theorem 5.1. The redundant sets are removed from the current level (line 7, cf. Proposition 5.3) then the next level can be generated (line 8) following the well-known Apriori technique. The loop is over when the new level is empty. The algorithm completes by displaying the CFDs discovered at the last valid level.

Algorithm CFun
1 $L_0 := < \overline{\emptyset}, 1, \overline{\emptyset}, \overline{\emptyset} >$
2 $L_1 := \{ < \overline{A}, |\overline{A}|, \overline{A}, \overline{A} > | \ \overline{A} \in ASP_{CFD}(R, r) \wedge |\overline{A}.att| = 1 \}$
3 **for** ($k := 1$; $L_k \neq \emptyset$; $k := k + 1$) **do**
4 ComputeClosures(L_{k-1}, L_k)
5 ComputeQuasiClosures(L_k, L_{k-1})
6 DisplayCFDs(L_{k-1})
7 PruneRedundantSets(L_k, L_{k-1})
8 L_{k+1} := GenerateCandidates(L_k)
9 ComputeClosures(L_{k-1}, L_k)
10 DisplayCFDs(L_{k-1})
end CFun

Example 5.1 *To illustrate the section (cf. Figure 3), we use the relation already described (cf. Figure 2). The first column of the following table is the* $\overline{X}$ *candidate which can be or not a conditional redundant set. The candidate prefixed by '*' is a conditional redundant set. The second column corresponds to the cardinality of* $\overline{X}$ *and the two last columns represent the conditional quasi-closure and conditional closure of* $\overline{X}$*. On the right, the CFDs discovered are displayed.*

Implementation technique We use the partitions representation introduced in [8, 26] and often used for the FD inference problem (cf. [14, 15, 19, 23, 24]). Indeed, one can compute quite efficiently the frequencies and generate only valid combinations of candidates (for instance, $\overline{AA}$ is invalid and will not be generated).

Example 5.2 *To illustrate the use of partitions, we again take the relation Figure 2. The threshold is set to 1. The partitions following the attributes A and C*

$\overline{X}$	$\lvert\overline{X}\rvert$	$\overline{X}^{\diamond}_{\Sigma}$	$\overline{X}^{*}_{\Sigma}$	
A0	3	A0	A0	
A2	2	A2	A2 C0 D1	$A2 \rightarrow C0D1$
B1	3	B1	B1	
B0	1	B0	A0 B0 C0 D1	$B0 \rightarrow A0C0D1$
B2	1	B2	A2 B2 C0 D1	$B2 \rightarrow A2C0D1$
C0	4	C0	C0	
C3	1	C3	A0 B1 C3 D2	$C3 \rightarrow A0B1D2$
D2	2	D2	A0 B1 D2	$D2 \rightarrow A0B1$
D1	3	D1	C0 D1	$D1 \rightarrow C0$
A0 B1	2	A0 B1	A0 B1 D2	$A0B1 \rightarrow D2$
A0 B0	1	A0 B0 C0 D1	A0 B0 C0 D1	
A2 B1	1	A2 B1 C0 D1	A2 B1 C0 D1	
A2 B2	1	A2 B2 C0 D1	A2 B2 C0 D1	
A0 C0	2	A0 C0	A0 C0	
A0 C3	1	A0 B1 C3 D2	AO B1 C3 D2	
A2 C0	2	A2 C0 D1	A2 C0 D1	
*A0 D2	2	A0 B1 D2	A0 B1 D2	
A0 D1	1	A0 C0 D1	A0 B0 C0 D1	$A0D1 \rightarrow B0$
A2 D1	2	A2 C0 D1	A2 C0 D1	
...	...	...	...	
*A0 B1 C0	1	A0 B1 C0 D2	A0 B1 C0 D2	
...	...	...	...	

Figure 3: Illustration of the proposed characterisation

are $\pi_A = \{(1,2,3),(4,5)\}$ *and* $\pi_C = \{(1,3,4,5),(2)\}$. *The values corresponded to equivalence classes are* 0, 2 *for* A *and* 0, 3 *for* C. *The product of* π_A *and* π_C *is* $\pi_{AC} = \{(1,3),(2),(4,5)\}$. *The values corresponded are* $(0,0)$, $(0,3)$ *and* $(2,0)$. *It directly provides the conditional attributes with their frequency:*
$freq(< A0 >) = 3$, $freq(< A2 >) = 2$, $freq(< C0 >) = 4$, $freq(< C3 >) = 1$, $freq(< A0, C0 >) = 2$, $freq(< A0, C3 >) = 1$, $freq(< A2, C0 >) = 2$.
Hence the CFD $A2 \rightarrow C0$ *is held since* $freq(< A2 >) = freq(< A2, C0) >) = 2$.
Moreover, no impossible combinations have been generated.

6 Experiments

In order to assess performances, the approach described in Section 5 has been implemented in C++. An executable file can be generated with Visual C++ 9.0 or GNU g++ compilers. Various experimentations have been performed on an Intel Pentium Centrino 2 GHz with 2 GB of main memory, running on Linux operating system. More details of our implementation and the source code are available at the following URL: http://pageperso.lif.univ-mrs.fr/~noel.novelli/CFDProject.

Source or executable code of [7, 11] have not been disclosed yet. To compare our proposition, we used the same real life datasets. The experiments used real datasets from the UCI machine learning repository (http://archive.ics.uci.edu/ml), namely the Winsconsin breast cancer (WBC) and Chess datasets (also used by others approaches). The following table summarises the real life datasets characteristics.

Datasets	#Attributes	#Tuples	Size (Ko)
Wirsconsin Breast Cancer	11	699	19 917
Chess	7	28 056	531 820

The Figure 4 shows the behavior of our approach applied on real life datasets when the threshold of frequent CFD varies. The curves on the left-hand side (resp. right-hand side) illustrate the execution time in seconds (memory usage in Mo). As expected, when the minimal support increases, the execution time and memory usage reduce.

We also generated synthetic data with our own random data generator: It is a generator of uniform data for each column independently of each other. The synthetic datasets are automatically generated using the following parameters: $|r|$ is the cardinality of the relation, $|R|$ stands for the number of attributes and c is the rate of correlation between attribute values. The more it increases, the more satisfied CFDs exist in the datasets.

The Figure 5 shows the behavior when the number of tuples goes from 5 000 to 50 000 on different synthetic datasets. The data correlation rate is set to 30% to access the scalability of our proposition in the number of tuples. Indeed, it allows us to fix the number of satisfied CFDs indepently of the number of tuples (for a frequency support set to 1). The memory usage and the execution time are linear according to the number of tuples.

The Figure 6 shows the behavior when the data correlation rates go from 30% to 70% on different synthetic datasets for a fixed number of tuples (set to 5 000) and a fixed number of attributes (set to 7). The idea is to study the behavior of our algorithm when the search space of conditional attributes grows.

The execution time and the memory usage increase slightly according to the data correlation rates which is a surprinsing result due to the inherent exponential complexity. The main reason comes from our very efficient implementation based on partitions of attribute values.

Moreover, it is worth noting that our implementation does not consume much memory ressources. For instance for a synthetic dataset with 1 000 000 tuples, and 9 attributes, the execution time is arround 31 seconds using 1 Gb of main memory. These results appear to be of the same order (in response time) than previous approaches [7, 11].

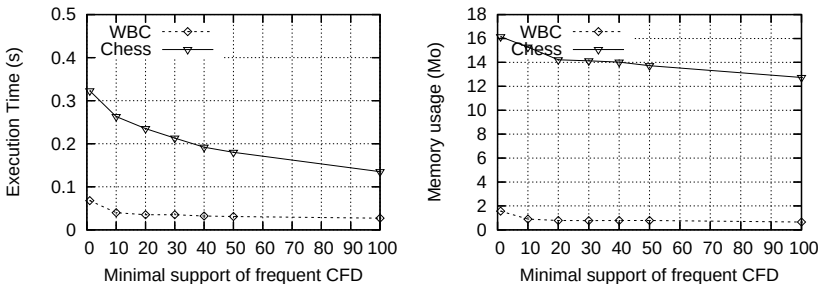

Figure 4: Execution time and memory usage for the *Wisconcin Breast Cancer* and *chess* real life datasets

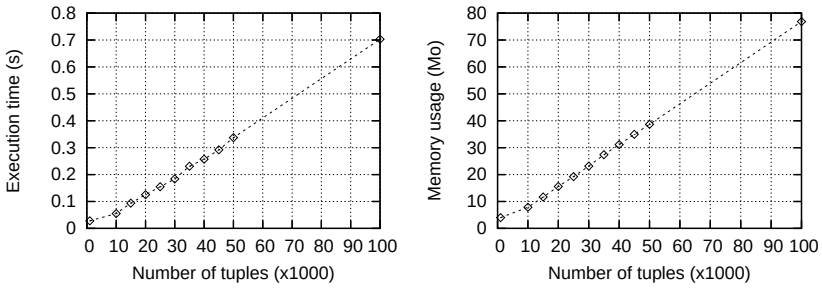

Figure 5: Execution time and memory usage for various number of tuples

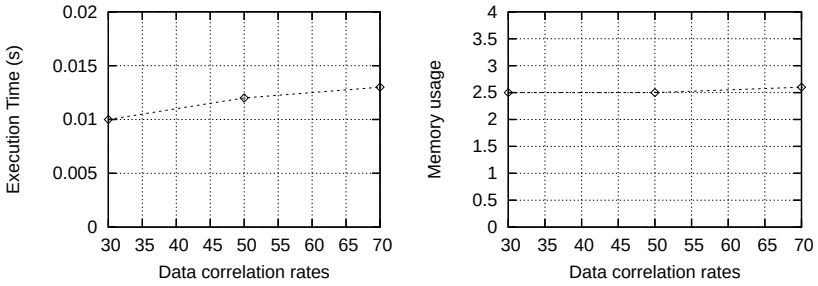

Figure 6: Execution time and memory usage for various data correlation rates

7 Conclusion

In this paper, we have studied the discovery of constant conditional functional dependencies in an existing relation. We have adapted two well-known approaches. The first one is based on *Conditional Agree Sets* and can be used to extract CFDs without frequency constraint. The theoretical machinery we have introduced is a contribution per se. The second one is an extension of the Fun approach [24] whose advantage is to easily deal with the frequency constraint. An efficient implementation has been proposed and tested against different synthetic and real-life datasets.

As future work, the implementation of the approach based on conditional agree sets has to be done, for example with the iZi library [12]. A thorough experimentation evaluation campaign remains to be done to assess our results. We also plan to address the problem of **variable** CFDs discovery. It is worth noting that mining frequent CFDs share some characteristics with the problem of mining frequent projection-selection conjunctive queries (see for example [17]). We plan to investigate the possible cross-fertilization between these two problems.

References

[1] Serge Abiteboul, Rick Hull, and Victor Vianu. *Fondements Des Bases De Données*. Vuibert, 2000.

[2] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993*, pages 207–216. ACM Press, 1993.

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

[4] Claude Berge. *Graphs and Hypergraphs*. North-Holland Mathematical Library 6. American Elsevier, 2d rev. ed. edition, 1976.

[5] Philip Bohannon, Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. Conditional functional dependencies for data cleaning. In *Proceedings of ICDE'07, April 15-20, Istanbul, Turkey*, pages 746–755, 2007.

[6] Paul De Bra and Jan Paredaens. Conditional dependencies for horizontal decompositions. In *Proceedings of the 10th Colloquium on Automata, Languages and Programming*, pages 67–82, London, UK, 1983. Springer-Verlag.

[7] Fei Chiang and Renée J. Miller. Discovering data quality rules. *PVLDB*, 1(1):1166–1177, 2008.

[8] S.S. Cosmadakis, P.C. Kanellakis, and N. Spyratos. Partition Semantics for Relations. *Journal of Computer and System Sciences*, 33(2):203–233, 1986.

[9] J. Demetrovics and V.D. Thi. Some remarks on generating Armstrong and inferring functional dependencies relation. *Acta Cybernetica*, 12(2):167–180, 1995.

[10] Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. Conditional functional dependencies for capturing data inconsistencies. *ACM Trans. Database Syst.*, 33(2), 2008.

[11] Wenfei Fan, Floris Geerts, Laks V. S. Lakshmanan, and Ming Xiong. Discovering conditional functional dependencies. In *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29 2009 - April 2 2009, Shanghai, China*, pages 1231–1234, 2009.

[12] Frédéric Flouvat, Fabien De Marchi, and Jean-Marc Petit. *Advanced Techniques for Data Mining and Knowledge Discovery*, chapter The iZi project: easy prototyping of interesting pattern mining algorithms, pages 1–15. LNCS. Springer-Verlag, September 2009.

[13] Lukasz Golab, Howard Karloff, Flip Korn, Divesh Srivastava, and Bei Yu. On generating near-optimal tableaux for conditional functional dependencies. *Proc. VLDB Endow.*, 1(1):376–390, 2008.

[14] Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen. Efficient Discovery of Functional and Approximate Dependencies Using Partitions. In *ICDE'98, Orlando, Florida, USA*, pages 392–401, 1998.

[15] Y. Huhtala, J. Karkkainen, P. Porkka, and H. Toivonen. TANE: An Efficient Algorithm for Discovering Functional and Approximate Dependencies. *The Computer Journal*, 42(2):100–111, 1999.

[16] Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen. Tane: An efficient algorithm for discovering functional and approximate dependencies. *The Computer Journal*, 42(3):100–111, 1999.

[17] Tao-Yuan Jen, Dominique Laurent, and Nicolas Spyratos. Mining all frequent projection-selection queries from a relational table. In *EDBT 2008, 11th International Conference on Extending Database Technology, Nantes, France, March 25-29, 2008, Proceedings*, pages 368–379, 2008.

[18] Jyrki Kivinen and Heikki Mannila. Approximate inference of functional dependencies from relations. *Theor. Comput. Sci.*, 149(1):129–149, 1995.

[19] Stéphane Lopes, Jean-Marc Petit, and Lotfi Lakhal. Efficient discovery of functional dependencies and armstrong relations. In *EDBT 2000*, volume 1777 of *LNCS*, pages 350–364, Konstanz, Germany, 2000. Springer.

[20] H. Mannila and K-J. Räihä. Algorithms for Inferring Functional Dependencies from Relations. *DKE*, 12:83–99, 1994.

[21] H. Mannila and Hannu Toivonen. Levelwise search and borders of theories in knowledge discovery. *DMKD*, 1(3):241–258, 1997.

[22] Raoul Medina and Lhouari Nourine. A unified hierarchy for functional dependencies, conditional functional dependencies and association rules. In *ICFCA*, Lecture Notes in Computer Science, pages 235–248. Springer, 2009.

[23] Noël Novelli and Rosine Cicchetti. Fun: An efficient algorithm for mining functional and embeddeddependencies. In *Proceedings of the 8th International Conference on DatabaseTheory (ICDT'01)*, volume 1973 of *Lecture Notes in Computer Science*, pages 189–203, 2001.

[24] Noël Novelli and Rosine Cicchetti. Functional and embedded dependency inference: a data mining point of view. *Information Systems (IS)*, 26(7):477–506, 2001.

[25] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering frequent closed itemsets for association rules. In *ICDT*, pages 398–416, 1999.

[26] Nicolas Spyratos. The partition model: A deductive database model. *ACM TODS*, 12(1):1–37, 1987.

[27] Catharine Wyss, Chris Giannella, and Edward Robertson. Fastfds: A heuristic-driven, depth-first algorithm for mining functional dependencies from relation instances extended abstract. *Data Warehousing and Knowledge Discovery*, pages 101–110, 2001.