Découverte des dépendances fonctionnelles conditionnelles fréquentes

Thierno Diallo* et Noël Novelli**

*Université de Lyon, LIRIS, CNRS-UMR5205
7 av. Jean Capelle, 69621 Villeurbanne Cedex, France
thierno.diallo@insa-lyon.fr
**Université de la Méditerranée, LIF, CNRS-UMR6166
Fac. des Sc. de Luminy, 163 av. de Luminy, 13288 Marseille Cedex 9, France
noel.novelli@lif.univ-mrs.fr

Résumé. Les Dépendances Fonctionnelles Conditionnelles (DFC) ont été introduites en 2007 pour le nettoyage des données. Elles peuvent être considérées comme une unification de Dépendances Fonctionnelles (DF) classiques et de Règles d'Association (RA) puisqu'elles permettent de spécifier des dépendances mixant des attributs et des couples de la forme attribut/valeur.
Dans cet article, nous traitons le problème de la découverte des DFC, i.e. déterminer une couverture de l'ensemble des DFC satisfaites par une relation r. Nous montrons comment une technique connue pour la découverte des DF (exactes et approximatives) peut être étendue aux DFC. Cette technique a été implémentée et des expériences ont été menées pour montrer la faisabilité et le passage à l'échelle de notre proposition.

Mots clés: Dépendances entre données, Fouille de données, Théorie des bases de données.

1 Introduction

Les précédents travaux sur l'amélioration de la qualité des données s'appuient principalement sur les classes de contraintes traditionnelles telles que les DF ou encore les DF Approximatives (Kivinen et Mannila (1995)). Même si les mesures g_1, g_2, et g_3 pour les DF Approximatives capturent certaines erreurs, leur expression n'est pas assez forte ou pas assez fine pour capturer les données incohérentes de manière précise. En effet, ces mesures permettent de détecter des erreurs sur les DF et non au niveau des classes de valeurs. Récemment, Bohannon et al. (2007) ont étendu les DF aux DFC pour pallier ce problème.

Dans cet article nous traitons le problème de l'inférence des DFC i.e. trouver une couverture de l'ensemble des DFC satisfaites dans une relation. Disposer de ces techniques de découverte permet entre autres d'améliorer la performance des outils de nettoyage de données basées sur les DFC. A notre connaissance deux contributions ont été faites sur la fouille de DFC (Chiang et Miller (2008); Fan et al. (2009)). Chiang et Miller (2008) présentent un outil

de gestion de la qualité des données en proposant des règles et en identifiant des tuples incohérents. Ils présentent aussi des méthodes pour la découverte de DFC satisfaites, cependant des DFC redondantes sont aussi générées. Fan et al. (2009) adaptent les méthodes bien connues pour la découverte des DF (TANE Huhtala et al. (1999b), FastFD Wyss et al. (2001)) pour inférer sur les DFC.

Contribution Dans cet article, nous introduisons nos travaux sur le problème de la découverte de DFC en utilisant la notion de partitions. Nous montrons que les techniques de fouilles de données proposées pour la découverte de dépendances fonctionnelles et de règles d'associations peuvent être réutilisées pour la découverte de DFC constantes et fréquentes. Nous montrons comment étendre une approche basée sur les notions d'ensembles libres, fermeture et quasi-fermeture pour l'inférence de DF à l'inférence de DFC. Nous avons implémenté cette approche et réalisé plusieurs expérimentations afin de prouver son efficacité lors du passage à l'échelle.

Organisation de l'article Dans la section 2, les notions de base concernant les DFC sont données et nous introduisons une nouvelle notation pour les DFC. La section 3 est consacrée à la découverte de DFC en offrant un cadre théorique et applicatif. Les expérimentations de découverte de DFC fréquentes se trouvent à la section 4. La dernière section est consacrée à la conclusion de ce travail ainsi qu'à ses perspectives.

2 Préliminaires

Les notations utilisées sont celles empruntées à Bohannon et al. (2007). Chaque attribut A a un domaine noté $DOM(A)$. Soit une relation r sur un schéma R avec $A \in R$, le domaine actif de A dans r est noté $ADOM(A, r)$.

Exemple 1 *Nous empruntons l'exemple fourni dans Bohannon et al. (2007) pour illustrer le concept de DFC. Soit* cust *un schéma de relation décrivant un client avec : country code (CC), area code (AC), phone number (PN), name (NM), street (STR), city (CT) and zip code (ZIP). La figure 1 illustre une relation, r_0, sur ce schéma. Soient $f_1 = CC, AC, PN \rightarrow STR, CT, ZIP$, $f_2 = CC, AC \rightarrow CT, ZIP$ et $f_3 = CC, ZIP \rightarrow STR$ trois DF. r_0 satisfait f_1 et f_2 mais viole f_3 (e.g. tuples t_4 et t_5). L'intuition derrière les DFC est d'extraire à travers une formule de sélection (égalité) un sous ensemble de la relation sur lequel on peut définir des DF. Par exemple la DF $f_3 : CC, ZIP \rightarrow STR$ pour $\sigma_{CC=44}(r_0)$ peut être exprimée à l'aide de la DFC $\phi_0 = (CC, ZIP \rightarrow STR, (44, \_ \parallel \_))$ où '$\parallel$' sépare la partie gauche de la partie droite et '_' représente n'importe quelle valeur du domaine de l'attribut correspondant.*

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

FIG. 1 – *Relation r_0 sur le schéma* cust

D'autres DFC comme $\phi_1 = (CC, AC, PN \rightarrow STR, CT, ZIP(01, 908, \_ \parallel \_, NYC, \_))$, $\phi_2 = (CC, AC, PN \rightarrow STR, CT, ZIP(01, 212, \_ \parallel \_, PHI, \_))$ *ou* $\phi_3 = (CC, AC \rightarrow CT(01, 215 \parallel PHI))$ *peuvent être définies sur* r_0.

Soit R un schéma de relation. Une DFC ρ sur R est une paire $(X \rightarrow Y, T_p)$ où $XY \subseteq R$, $X \rightarrow Y$ une DF et T_p un *pattern tableau* d'attributs de R.

Pour chaque $A \in R$ et pour chaque pattern tuple $t_p \in T_p$, $t_p[A]$ est soit une constante $\in DOM(A)$, soit la "variable sans nom" notée '_' si $A \in XY$, soit la "variable vide" notée '$*$' qui indique que l'attribut correspondant ne contribue pas au pattern (i.e. $A \in R - XY$).
Soit r une relation sur R, $X \subseteq R$ et T_p un pattern tableau sur R. Un tuple $t \in r$ *matche* un tuple $t_p \in T_p$ sur X, noté $t[X] \asymp t_p[X]$, ssi $\forall A \in X$, $t[A] = t_p[A]$, ou $t_p[A] =$'_', ou $t_p[A] =$'$*$'.
Soit r une relation sur R et $\rho = (X \rightarrow Y, T)$ une DFC avec $XY \subseteq R$:

- r satisfait ρ, noté $r \models \rho$, ssi $\forall t_i, t_j \in r$ et $\forall t_p \in T$, si $t_i[X] = t_j[X] \asymp t_p[X]$ alors $t_i[Y] = t_j[Y] \asymp t_p[Y]$.
- r viole la DFC $\rho = (X \rightarrow Y, T)$, notée $r \not\models \rho$, ssi
  - il existe un tuple $t \in r$ et un pattern tuple $t_p \in T$ tel que $t[X] \asymp t_p[X]$ et $t[Y] \not\asymp t_p[Y]$ ou
  - il existe $t_i, t_j \in r$ et un pattern tuple $t_p \in T$ tel que $t_i[X] = t_j[X] \asymp t_p[X]$ et $t_i[Y] \neq t_j[Y]$.

Exemple 2 *La relation* r_0 *(cf. figure 1) satisfait les DFC* ϕ_0, ϕ_1 *et* ϕ_3 *alors qu'elle ne satisfait pas les deux DFC* ϕ_2 *et* ϕ_4 *suivantes :*

- $\phi_2 = (CC, AC, PN \rightarrow STR, CT, ZIP(01, 212, \_ \parallel \_, PHI, \_))$ *car le tuple* t_3 *viole* ϕ_2 : $t_3\,[CC, AC, PN] \asymp (01, 212, \_)$ *mais* $t_3\,[STR, CT, ZIP] \not\asymp (\_, PHI, \_)$.
- $\phi_4 = ([CC, CT] \rightarrow ZIP, (01, \_ \parallel \_))$ *car* t_2 *et* t_3 *violent* ϕ_4 *puisque* $t_2\,[CC, CT] = t_3\,[CC, CT] \asymp (01, \_)$, *mais* $t_2\,[ZIP] \neq t_3\,[ZIP]$.

Une DFC $(X \rightarrow Y, T_p)$ est dite à la *forme normale* (Fan et al. (2008)), lorsque $|Y| = 1$ et $|T_p| = 1$. Par la suite nous n'étudions que les DFC à la forme normale.

Une DFC $(X \rightarrow A, t_p)$ est dite :

- *constante* si $t_p\,[XA]$ n'est constitué que de constantes.
- *variable* si la partie droite de son pattern tuple est une variable sans nom.

3 Découverte des DFC constantes et fréquentes

Nous souhaitons découvrir les DFC constantes et fréquentes présentes dans une relation, i.e. les DFC pour lesquelles le nombre de tuples de la relation les satisfaisant est supérieur à un seuil de fréquence donné.

Pour cela, nous établissons un cadre théorique offrant une caractérisation simple et solide sur laquelle s'appuie une implémentation efficace utilisant la notion de partition.

3.1 Cadre théorique

Nous commençons par définir les notations nécessaires pour caractériser l'espace de recherche et l'expression de DFC puis nous poursuivons par une caractérisation permettant la découverte de DFC dans une relation existante.

Définition 1 *Soit R un schéma de relation. L'espace de recherche des DFC constantes sur R, noté $SP_{CFD}(R)$, est défini comme suit :*

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

Ainsi toute DFC constante peut être représentée à l'aide des éléments de cet espace. Soit $\rho = (A_1 \dots A_n \rightarrow A, t_p[A_1 \dots A_n A])$ une DFC constante sur R. ρ est équivalent à $\overline{X} \rightarrow \overline{A}$, avec $\overline{X} = \{(A_1, t_p[A_1]), \dots, (A_n, t_p[A_n])\} \subseteq SP_{CFD}(R)$ et $\overline{A} = (A, t_p[A]) \in SP_{CFD}(R)$.
Soit $\overline{X} \subseteq SP_{CFD}(R)$, $\overline{X}.att$ représente l'union de tous les attributs appartenant à $\overline{X}$.

Définition 2 *Soit R un schéma de relation et r une relation sur R. L'espace de recherche des DFC constantes pour r, noté $ASP_{CFD}(R, r)$, est défini comme suit :*

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

Les éléments de $ASP_{CFD}(R, r)$ sont appelés des *ensembles d'attributs conditionnels*.

Exemple 3 *Soit r la relation de la figure 2. Nous notons le couple (A_i, v) par $A_i v$.*
$ASP_{CFD}(ABCD, r) = \{A0, A2, B0, B1, B2, C0, C3, D1, D2\}$

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

FIG. 2 – *Une relation r sur $R = ABCD$*

Une DFC $\overline{X} \rightarrow \overline{A}$ sur R est dite *triviale* si $\overline{A}.att \in \overline{X}.att$. Dans notre étude nous ne considérons que les DFC non triviales. Une DFC constante $\overline{X} \rightarrow \overline{A}$ est *réduite à gauche* sur r si $\forall\, \overline{X'}.att \subset \overline{X}.att$, $r \not\models \overline{X'} \rightarrow \overline{A}$. Une *DFC minimale* (Fan et al. (2008)) ρ sur r est non triviale, réduite à gauche et telle que $r \models \rho$. Une *couverture canonique* d'un ensemble Σ_r de DFC est un ensemble Σ_{cc} de DFC minimales tel que Σ_r est équivalent à Σ_{cc}.

Etant donné une relation r notre problème est d'énumérer la couverture canonique des DFC satisfaites par r.

La propriété suivante montre la monotonie des DFC ce qui correspond à un ordre partiel.

Propriété 1 *Soit r une relation sur $R, \overline{X}, \overline{Y} \subseteq ASP_{CFD}(R, r)$ tel que $\overline{X} \subseteq \overline{Y}$ et $\overline{A} \in ASP_{CFD}(R, r)$. Nous avons : $r \models \overline{X} \rightarrow \overline{A} \Rightarrow r \models \overline{Y} \rightarrow \overline{A}$.*

A l'aide de ces nouvelles notations, la fermeture d'un ensemble d'attributs conditionnels définie dans Fan et al. (2008) peut être facilement ré-écrite.

Définition 3 *Soit Σ une DFC constante et $\overline{X} \subseteq SP_{CFD}(R)$. La fermeture de $\overline{X}$ sur Σ, notée $\overline{X}^*_\Sigma$, est définie comme suit : $\overline{X}^*_\Sigma = \{\overline{A} \in SP_{CFD}(R) \mid \Sigma \models \overline{X} \rightarrow \overline{A}\}$.*

L'opérateur $.^*_\Sigma$ défini sur l'ensemble des parties de $SP_{CFD}(R)$ est un opérateur de fermeture, i.e. extensible, monotone et idempotent.

Intuitivement, la fréquence d'une DFC dans une relation est le nombre de tuples qui satisfont le pattern tuple, i.e. la taille de la requête de sélection correspondante.

Définition 4 *Soit $\theta = (\overline{X} \rightarrow \overline{Y})$ une DFC constante sur R et r une relation sur R. La fréquence de θ dans r, notée $freq(\theta, r)$, est définie comme suit :*

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

Soit ϵ un entier représentant la valeur d'un seuil. Une DFC θ est dite fréquente dans r, si $freq(\theta, r) \geq \epsilon$.

Cette définition est importante pour éliminer ou réduire l'impact des données impropres contenues dans une relation. En effet, seules les données suffisamment présentes sont prises en compte ce qui gomme par exemple les erreurs de saisie ce qui correspond à la même philosophie que la mesure g_3 introduite dans Kivinen et Mannila (1995).
Bien évidemment, ce prédicat est monotone ce qui entraine la propriété suivante.

Propriété 2 *Soit r une relation sur $R, \overline{X}, \overline{Y} \subseteq ASP_{CFD}(R, r)$ tels que $\overline{X} \subseteq \overline{Y}$ et ϵ un seuil. Nous avons : $freq(\overline{Y}, r) \geq \epsilon \Rightarrow freq(\overline{X}, r) \geq \epsilon$ (ou $freq(\overline{X}, r) < \epsilon \Rightarrow freq(\overline{Y}, r) < \epsilon$)*

Nous avons besoin de définir un test afin de décider si une DFC est valide ou non dans une relation. Depuis longtemps, nous savons qu'une telle propriété existe pour tester la satisfaction d'une DF dans une relation. La propriété suivante est souvent utilisée : $r \models X \rightarrow Y$ ssi $|X|_r = |XY|_r$.
Pour les DFC, une propriété similaire peut être obtenue. C'est ce que nous montrons ci-après en utilisant l'opérateur de sélection de l'algèbre relationnelle.

Propriété 3 *Soit R un symbole de relation, r une relation sur R, $\overline{X}, \overline{Y} \subseteq ASP_{CFD}(R, r)$ et $C_{\overline{X}}, C_{\overline{Y}}$ deux formules de sélection sur $\overline{X}$ et $\overline{Y}$ respectivement.*
$r \models \overline{X} \rightarrow \overline{Y}$ ssi $|\sigma_{C_{\overline{X}}}(r)| = |\sigma_{C_{\overline{X}} \wedge C_{\overline{Y}}}(r)|$ où $C_{\overline{X}} = \wedge_{(A,v)\in\overline{X}}(A = v)$ et $C_{\overline{Y}} = \wedge_{(A,v)\in\overline{Y}}(A = v)$

Nous pouvons maintenant introduire les définitions qui nous permettront d'établir une caractérisation solide permettant la découverte de DFC.

Définition 5 *Ensembles conditionnels libres*
Soit $\overline{X} \subseteq ASP_{CFD}(R, r)$ un ensemble d'attributs conditionnels.
$\overline{X}$ est un ensemble conditionnel libre dans r ssi $\nexists \overline{X'} \subset \overline{X}$ tel que $|\sigma_{C_{\overline{X'}}}(r)| = |\sigma_{C_{\overline{X}}}(r)|$.

L'ensemble des ensembles conditionnels libres dans r est noté $\mathcal{CFS}_r$. Tout ensemble d'attributs conditionnels qui n'est pas inclus dans $\mathcal{CFS}_r$ est appelé un *ensemble conditionnel non libre*.

Propriété 4 *Soit r une relation sur $R, \overline{X}, \overline{Y} \subseteq ASP_{CFD}(R, r)$ tel que $\overline{X} \subseteq \overline{Y}$. Nous avons : $\overline{Y} \in \mathcal{CFS}_r \Rightarrow \overline{X} \in \mathcal{CFS}_r$ (ou de façon équivalente $\overline{X} \notin \mathcal{CFS}_r \Rightarrow \overline{Y} \notin \mathcal{CFS}_r$)*

A partir des propriétés et des définitions introduites dans cette section, il est évident que les algorithmes par niveau (comme *Apriori*) peuvent être utilisés pour découvrir les ensembles conditionnels libres (conjonction de deux prédicats monotones). A partir des ensembles conditionnels libres, nous étendons les résultats donnés dans Novelli et Cicchetti (2001a,b) (pour l'inférence de DF) pour proposer une nouvelle caractérisation de la couverture canonique des DFC qui tient compte d'un seuil de fréquence. Elle est basée sur les ensembles conditionnels libres, leur fermeture et leur quasi-fermeture.

Définition 6 *Fermeture d'un ensemble d'attributs conditionnels dans une relation*
Soit $\overline{X}$ un ensemble d'attributs conditionnels, $\overline{X} \subseteq ASP_{CFD}(R, r)$. Sa fermeture dans r est définie comme suit : $\overline{X}^*_{\Sigma_r} = \overline{X} \cup \{\overline{A} \mid \overline{A}.att \in R - \overline{X}.att \wedge |\sigma_{C_{\overline{X}}}(r)| = |\sigma_{C_{\overline{X}} \wedge C_{\overline{A}}}(r)|\}$.

Définition 7 *Quasi-fermeture d'un ensemble d'attributs conditionnels dans une relation*
La quasi-fermeture d'un ensemble d'attributs conditionnels $\overline{X} \subseteq ASP_{CFD}(R, r)$, noté $\overline{X}^\diamond_{\Sigma_r}$, est définie par :

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

Grâce à la propriété de monotonie de l'opérateur de fermeture, nous avons : $\overline{X} \subseteq \overline{X}^\diamond_{\Sigma_r} \subseteq \overline{X}^*_{\Sigma_r}$ ce qui nous permet de réécrire la définition 6 de la façon suivante.

Définition 8 *Fermeture d'un ensemble d'attributs conditionnels dans une relation*
Soit $\overline{X}$ un ensemble d'attributs conditionnels, $\overline{X} \subseteq ASP_{CFD}(R, r)$. Sa fermeture dans r est redéfinie comme suit : $\overline{X}^*_{\Sigma_r} = \overline{X}^\diamond_{\Sigma_r} \cup \{\overline{A} \mid \overline{A}.att \in R - \overline{X}^\diamond.att \wedge |\sigma_{C_{\overline{X}}}(r)| = |\sigma_{C_{\overline{X}} \wedge C_{\overline{A}}}(r)|\}$.

La notion de quasi-fermeture est utilisée pour accumuler les fermetures des sous ensembles propres de l'attribut conditionnel considéré. Ceci nous permet dans le théorème suivant de garantir la minimalité des DFC extraites.
Le théorème suivant prouve que l'ensemble de DFC constantes caractérisé à l'aide des concepts introduits est la couverture canonique des DFC constantes pour la relation r.

Théorème 1 $\Sigma_{cc}(r, \epsilon) = \{\overline{X} \rightarrow \overline{A} \mid \overline{X} \in \mathcal{CFS}_r, freq(\overline{X}, r) \geq \epsilon$ *et* $\overline{A} \in \overline{X}^*_{\Sigma_r} - \overline{X}^\diamond_{\Sigma_r}\}$

La fréquence $freq(\overline{X}, r)$ étant utilisée pour filtrer les DFC fréquentes, elle n'a pas à intervenir dans la démonstration.

Preuve 1 *Supposons tout d'abord que la DFC $\overline{X} \rightarrow \overline{A} \in CC(r, \epsilon)$ (DFC minimale et non triviale). Nous avons donc $\forall\ \overline{X}' \subset \overline{X}$, $|\overline{X}'|_r \neq |\overline{X}|_r$ donc $\overline{X} \in \mathcal{CFS}_r$ (conformément à la définition 5). De plus, $\overline{A} \in \overline{X}^*_{\Sigma_r}$ puisque la DFC est valide. Elle est minimale donc $\overline{A} \notin \overline{X}^\diamond_{\Sigma_r}$ (cf. Définition 8). Donc si la DFC $\overline{X} \rightarrow \overline{A} \in CC(r, \epsilon)$ alors $\overline{X} \in \mathcal{CFS}_r$ et $\overline{A} \in \overline{X}^*_{\Sigma_r} - \overline{X}^\diamond_{\Sigma_r}$. Supposons maintenant que $\overline{X} \in \mathcal{CFS}_r$ et $\overline{A} \in \overline{X}^*_{\Sigma_r} - \overline{X}^\diamond_{\Sigma_r}$. Puisque $\overline{A} \in \overline{X}^*_{\Sigma_r}$, $\overline{A}$ est déterminée par $\overline{X}$, donc $r \models \overline{X} \rightarrow \overline{A}$. Elle est minimale puisque $\overline{X} \in \mathcal{CFS}_r$ donc $\forall\ \overline{X}' \subset \overline{X}$, $|\overline{X}'|_r \neq |\overline{X}|_r$ et non triviale puisque $\overline{A} \notin \overline{X}^\diamond_{\Sigma_r}$. Donc $\overline{X} \rightarrow \overline{A} \in CC(r, \epsilon)$.*

Exemple 4 *Pour illustrer cette section (cf. figure 3), nous utilisons la relation déjà décrite (cf. figure 2). La première colonne du tableau suivant représente le candidat $\overline{X}$ qui peut être un ensemble conditionnel libre ou non libre. Les candidats précédés d'un '*' sont des ensembles conditionnels non libres. La deuxième colonne correspond à la cardinalité de $\overline{X}$ et pour finir, les deux dernières colonnes représentent la quasi-fermeture conditionnelle ($\overline{X}^\diamond_\Sigma$) et la fermeture conditionnelle ($\overline{X}^*_\Sigma$) de $\overline{X}$. A droite, les DFC découvertes sont affichées.*

$\overline{X}$	$\lvert\overline{X}\rvert$	$\overline{X}^{\diamond}_{\Sigma}$	$\overline{X}^{*}_{\Sigma}$	
A0	3	A0	A0	
A2	2	A2	A2 C0 D1	A → CD (2 || 0, 1)
B1	3	B1	B1	
B0	1	B0	A0 B0 C0 D1	B → ACD (0 || 0, 0, 1)
B2	1	B2	A2 B2 C0 D1	B → ACD (2 || 2, 0, 1)
C0	4	C0	C0	
C3	1	C3	A0 B1 C3 D2	C → ABD (3 || 0, 1, 2)
D2	2	D2	A0 B1 D2	D → AB (2 || 0, 1)
D1	3	D1	C0 D1	D → C (1 || 0)
A0 B1	2	A0 B1	A0 B1 D2	AB → D (0, 1 || 2)
*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	AD → B (0, 1 || 0)
A2 D1	2	A2 C0 D1	A2 C0 D1	
...	...	...	...	

FIG. 3 – *Illustration de la caractérisation proposée*

3.2 Implémentation de l'approche

Le cadre théorique proposé est très bien adapté à une approche par niveau pour l'implémentation de la découverte de DFC. Notre algorithme, appelé CFUN, est donc basé sur les concepts d'*Apriori* afin de découvrir l'ensemble des ensembles conditionnels libres. Une fois ces derniers découverts pour chaque niveau ainsi que les valeurs correspondantes de fréquence (comptage), quasi-fermeture et fermeture, la découverte des DFC est triviale suivant le théorème 1.
Cette philosophie est la même que celle utilisée pour l'extraction de DF de l'approche FUN de Novelli et Cicchetti (2001a,b).

Algorithme CFUN L'algorithme général, donné ci-après, initialise les niveaux L_0 (ligne 1) et L_1 (ligne 2) avant de commencer à proprement dit la boucle de calcul niveau par niveau (lignes 3-8). Cette dernière calcule tout d'abord la fermeture (ligne 4) de tous les attributs conditionnels du niveau L_{k-1} à l'aide du niveau L_k (comparaison de fréquences) puis calcule les quasi-fermetures (ligne 5) du niveau L_k à partir des fermetures obtenues au niveau précédant L_{k-1} (Définition 6). Une fois les fermetures et quasi-fermetures du niveau L_{k-1} calculées il est trivial d'afficher les DFC fréquentes obtenues à ce niveau (Théorème 1). Une phase d'élagage élimine les éléments du niveau courant qui n'appartiennent pas à $\mathcal{CFS}_r$ (ligne 7). Ceci permet lors de la génération des candidats du niveau suivant (ligne 8) que seuls les sur-ensembles des ensembles conditionnels libres soient considérés. L'algorithme se termine

en affichant les DFC découvertes au dernier niveau (ligne 9).
Chaque niveau contient des quadruplets < $\overline{X}$, $|\overline{X}|$, $\overline{X}^{\diamond}_{\Sigma}$, $\overline{X}^{*}_{\Sigma}$ > comme le montre la figure 3.

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 ComputeClosure(L_{k-1}, L_k)
5 ComputeQuasiClosure(L_k, L_{k-1})
6 DisplayCFD(L_{k-1})
7 PruneNonFreeSets(L_k, L_{k-1})
8 L_{k+1} := GenerateCandidate(L_k)
9 DisplayCFD(L_{k-1})
end CFUN

La phase de génération de candidats requiert une attention toute particulière. En effet, il ne faut pas générer de combinaison d'attributs conditionnels qui n'existe pas dans la relation considérée (par exemple, la combinaison $\overline{AA}$). Pour éviter ces erreurs de génération de candidats, nous avons utilisé la notion de partition.

Optimisation et ressource mémoire nécessaire Afin de pallier le problème de génération et d'optimiser les calculs, nous avons opté pour une structure interne de représentation de l'information basée sur la notion de partitions introduite par Cosmadakis et al. (1986); Spyratos (1987). L'opération de partitionnement d'une relation r selon un ensemble d'attributs X consiste à regrouper en classes d'équivalence, les tuples ayant même valeur pour X. L'ensemble des classes d'équivalence constitue une *partition* de r selon X. Les partitions peuvent être représentées de deux manières efficaces d'après Novelli et Maabout (2003) : soit on utilise un vecteur de numéros de classe d'équivalence dont l'indexation se fait par le numéro du tuple ; soit on utilise un vecteur d'identifiants de tuples rangés dans l'ordre des classes d'équivalence ce qui est très bien adapté aux calculs de DF (cf. Huhtala et al. (1998, 1999a); Lopes et al. (2000); Novelli et Cicchetti (2001a,b)) et donc pour les DFC. De plus, le produit de partitions proposé dans Novelli et Maabout (2003) nous permet aussi de calculer très rapidement les fréquences de chaque attributs conditionnels en ne générant pas les combinaisons inexistantes dans la relation.

Exemple 5 *Pour illustrer l'utilisation des partitions, nous reprenons la relation de la figure 2. Le seuil est fixé à 1. Les partitions suivant les attributs A et C sont $\pi_A = \{(1, 2, 3), (4, 5)\}$ et $\pi_C = \{(1, 3, 4, 5), (2)\}$. Les valeurs correspondantes aux classes d'équivalence sont 0, 2 pour A et 0, 3 pour C. Le produit de π_A et π_C est $\pi_{AC} = \{(1, 3), (2), (4, 5)\}$. Les valeurs correspondantes sont $(0, 0)$, $(0, 3)$ et $(2, 0)$. Ceci nous fournit directement les attributs conditionnels avec leur fréquence : $freq(< A0 >) = 3$, $freq(< A2 >) = 2$, $freq(< C0 >) = 4$, $freq(< C3 >) = 1$, $freq(< A0, C0 >) = 2$, $freq(< A0, C3 >) = 1$, $freq(< A2, C0 >) = 2$. D'où la DFC $A \rightarrow C(2||0)$ est valide puisque $freq(< A2 >) = freq(< A2, C0) >) = 2$. De plus, aucune combinaison impossible n'a été générée.*

Pour une relation à n tuples et k attributs, le nombre correspondant de combinaisons d'attributs est de 2^k (ensemble des parties). Dans le pire des cas, chaque combinaison d'attributs engendre n combinaisons d'attributs conditionnels (toutes les valeurs de cette combinaison d'attributs sont distinctes). Cela implique que dans le pire des cas le nombre total d'attributs conditionnels est de $n \times 2^k$. Chaque attribut conditionnel est représenté par un quadruplet d'entier ($4 \times$

4 octets) ce qui signifie que la quantité maximale de mémoire pour conserver la totalité des niveaux de notre approche est de $16n \times 2^k$ octets.
L'algorithme proposé étant par niveau, seuls deux niveaux sont obligatoirement conservés en mémoire. De fait, les besoins mémoire sont très facilement réduits à $16n \times 2\binom{k}{k/2}$ octets soit à la partie la plus large du treillis de combinaisons d'attributs (les deux niveaux centraux).

4 Expérimentations

Afin d'évaluer les performances, l'approche décrite dans la section 3.1 a été implémentée en C++. L'exécutable peut être obtenu à l'aide des compilateurs MS Visual C++ 9.0 ou GNU g++. Différentes expérimentations ont été réalisées sur un ordinateur équipé d'un processeur Pentium Centrino 2 GHz avec 2 Go de RAM avec un système Linux.

Au moment de la rédaction de l'article, les auteurs des approches existantes (Chiang et Miller (2008); Fan et al. (2009)) n'ont pas encore pu nous faire parvenir un exécutable ou le code source de leurs approches. De fait, nous n'avons pas pu comparer notre proposition avec les leurs. Toutefois, nous avons utilisé les mêmes jeux de données réelles ce qui nous permet d'effectuer une comparaison approximative grossière des résultats. Pour les expérimentations sur des données réelles, nous avons utilisé Winsconsin breast cancer (WBC) et Chess datasets (comme les approches citées) issues du *UCI machine learning repository*[1].
Le tableau suivant résume les caractéristiques des jeux de données réelles.

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

La figure 4 montre le comportement de notre approche sur les données réelles décrites précédemment lorsque le seuil de fréquence des DFC varie. La courbe de gauche représente les temps d'exécution en seconde alors que celle de droite illustre les consommations mémoire en Mo. Comme attendu, quand le support minimal augmente, le temps d'exécution et la consommation mémoire diminuent.

Nous avons aussi généré des données synthétiques avec notre propre générateur de données : c'est un générateur uniforme de données pour chaque colonne indépendamment des autres colonnes de la relation. Les jeux de données synthétiques sont automatiquement générés à l'aide des paramètres suivants : $|r|$ représente la cardinalité de la relation, $|R|$ correspond au degré de la relation et c fixe le taux de corrélation entre les valeurs d'un attribut. Plus ce taux est grand, plus le nombre de DFC satisfaites dans la relation augmente.

La figure 5 illustre le comportement lorsque le nombre de tuples augmente de 5 000 à 50 000 pour différents jeux de données synthétiques. Le taux de corrélation de données est fixé à 30% pour montrer le passage à l'échelle de notre proposition en fonction du nombre de tuples. En effet, cela permet de fixer le nombre de DFC satisfaites indépendamment du nombre de tuples. Le support est fixé à 1 afin qu'aucune technique d'élagage ne soit appliquée ce qui correspond au pire cas d'extraction de DFC. Les besoins mémoire et le temps d'exécution sont linéaires au nombre de tuples de la relation considérée.

La figure 6 illustre le comportement de notre approche lorsque le taux de corrélation de données varie de 30% à 70% sur différents jeux de données synthétiques avec un nombre fixe

[1] `http ://archive.ics.uci.edu/ml`

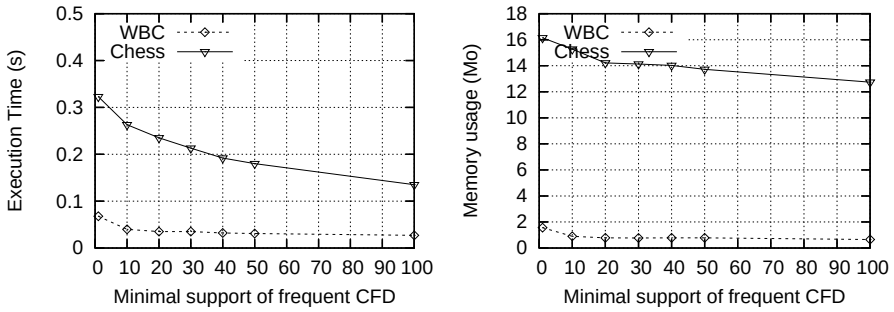

FIG. 4 – *Temps d'exécution et consommation mémoire pour les jeux de données réelles* Wisconcin Breast Cancer *et* chess

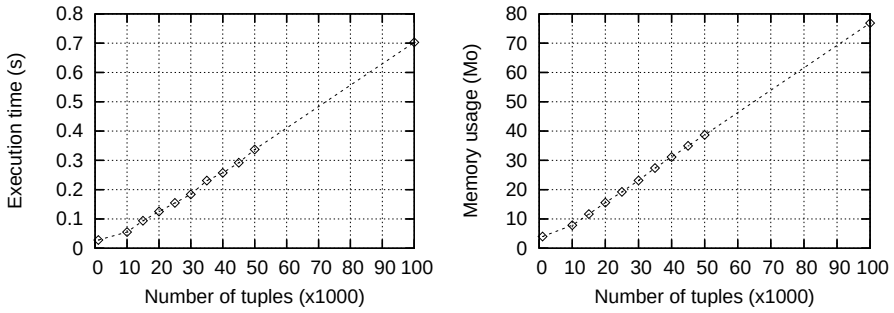

FIG. 5 – *Temps d'exécution et consommation mémoire pour différentes cardinalités*

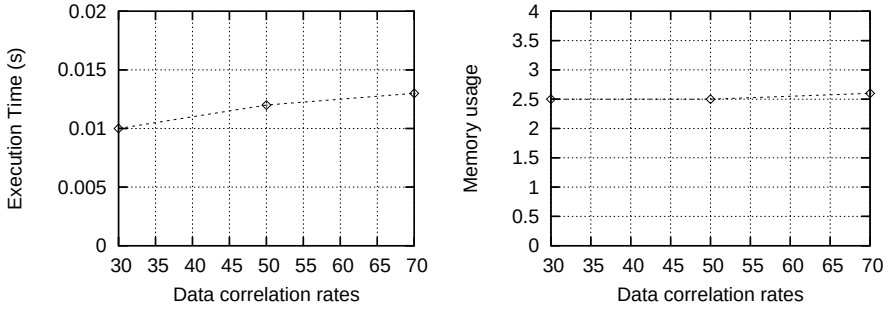

FIG. 6 – *Temps d'exécution et besoin mémoire pour différents taux de corrélation*

de tuples (5 000) et d'attributs (7). L'idée est d'étudier le comportement de notre algorithme lorsque la taille de l'espace de recherche des attributs conditionnels croît.

Le temps d'exécution et les besoins mémoire augmentent légèrement en fonction du taux de corrélation ce qui est un résultat surprenant en raison de la complexité exponentielle inhérente. La principale raison est due à l'efficacité de notre implémentation basée sur les partitions

de valeurs des attributs et leur produit (Novelli et Maabout (2003)).

En outre, il est intéressant de noter que notre mise en œuvre ne consomme que peu de mémoire. Par exemple, pour un ensemble de données synthétiques avec 1 000 000 de tuples et 9 attributs, le temps d'exécution approche 31 secondes en n'utilisant que 1 Go de mémoire principale.

5 Conclusion

Dans cet article, nous introduisons une nouvelle notation pour les dépendances fonctionnelles conditionnelles rendant leur découverte à partir d'une relation existante plus facile. Nous avons proposé l'adaptation d'une approche bien connue dans le contexte de l'inférence de dépendances fonctionnelles à la découverte de DFC constantes fréquentes. Elle est basée sur l'approche FUN (Novelli et Cicchetti (2001a,b)) et peut être utilisée pour l'extraction de DFC constantes fréquentes. Cette approche a été implémentée et testée sur différents jeux de données synthétiques et réelles.

Par souci de clarté, nous mettons l'accent dans cet article sur la découverte de DFC constantes. Dans l'avenir, nous envisageons de traiter le problème de la découverte de DFC variables. Deux approches sont en effet possibles : la première consiste à réduire l'ensemble des DFC constantes (si toutes les valeurs de $ADOM(X, r)$ sont présentes alors nous pouvons les substituer par '_') pour obtenir un ensemble équivalent restreint de DFC variables. La seconde consiste à enrichir l'approche actuelle pour obtenir des algorithmes dédiés.

Une campagne approfondie d'expérimentations reste à faire pour évaluer en profondeur nos résultats actuels.

Il est à noter que l'extraction de DFC fréquentes partage certaines caractéristiques avec le problème d'extraction de requêtes conjonctives fréquentes de projection-sélection (voir par exemple Jen et al. (2008)). Nous avons l'intention d'approfondir ces deux domaines afin d'en extraire de nouvelles techniques bénéfiques à leurs problèmes.

Références

Bohannon, P., W. Fan, F. Geerts, X. Jia, et A. Kementsietsidis (2007). Conditional functional dependencies for data cleaning. In *Proceedings of ICDE'07, April 15-20, Istanbul, Turkey*, pp. 746–755.

Chiang, F. et R. J. Miller (2008). Discovering data quality rules. *VLDB 1*(1), 1166–1177.

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

Fan, W., F. Geerts, X. Jia, et A. Kementsietsidis (2008). Conditional functional dependencies for capturing data inconsistencies. *ACM Trans. Database Syst. 33*(2).

Fan, W., F. Geerts, L. V. S. Lakshmanan, et M. Xiong (2009). Discovering conditional functional dependencies. In *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29 2009 - April 2 2009, Shanghai, China*, pp. 1231–1234.

Huhtala, Y., J. Kärkkäinen, P. Porkka, et H. Toivonen (1998). Efficient Discovery of Functional and Approximate Dependencies Using Partitions. In *ICDE'98, Orlando, Florida, USA*, pp. 392–401.

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

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

Jen, T.-Y., D. Laurent, et N. Spyratos (2008). 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*, pp. 368–379.

Kivinen, J. et H. Mannila (1995). Approximate inference of functional dependencies from relations. *Theor. Comput. Sci. 149*(1), 129–149.

Lopes, S., J. Petit, et L. Lakhal (2000). Efficient Discovery of Functional Dependencies and Armstrong Relations. In *Proc. EDBT'00*, pp. 350–364.

Novelli, N. et R. Cicchetti (2001a). Fun : An efficient algorithm for mining functional and embeddeddependencies. In *Proceedings of the 8th International Conference on Database-Theory (ICDT'01)*, Volume 1973 of *Lecture Notes in Computer Science*, pp. 189–203.

Novelli, N. et R. Cicchetti (2001b). Functional and embedded dependency inference : a data mining point of view. *Information Systems (IS) 26*(7), 477–506.

Novelli, N. et S. Maabout (2003). Algorithme efficace de calcul du produit de partitions et ses applications. In *19ème conférence Bases de Données Avancées (BDA'03)*, pp. 343–362.

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

Wyss, C., C. Giannella, et E. Robertson (2001). Fastfds : A heuristic-driven, depth-first algorithm for mining functional dependencies from relation instances extended abstract. *Data Warehousing and Knowledge Discovery*, 101–110.

Summary

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 ongoing work 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 how to extent a know technique for discovering exacts and approximates FDs to discovering CFDs. We have implemented the technique on which experiments have been carried out showing both the feasibility and the scalability of our proposition.

Keywords: Data dependencies, Data mining, Databases theory.