REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE

LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOULOUD MAMMERI TIZI-OUZOU

FACULTE DE GENIE ELECTRIQUE ET INFORMATIQUE

DEPARTEMENT D'INFORMATIQUE

Memoire

DE FIN D'ETUDES

DE MASTER ACADEMIQUE

Domaine : Mathématiques et Informatiques

Filière : Informatique

Spécialité : Conduite de Projets Informatiques

Thème : Analyse de concepts formels pour des implications d'attributs possibles.

Proposé et dirigé par :

M^me^ Ait-Yakoub Zina.

Réalisé par :

Mr Saal Aghiles.

Devant le jury d'examen composé de :

Mr	HAMMACHE Arezki	MCB	UMMTO	Président
Mr	RADJA Hakim	MAA	UMMTO	Examinateur
Mr	SAIDANI Rédha Fayçal	MAB	UMMTO	Examinateur

Promotion : 2018/2019.

Remerciements

Mes vifs remerciements accompagnés de toute ma gratitude vont à mon encadreur Madame Z. Ait Yakoub.

Ses conseils et ses encouragements ont permis à ce travail d'aboutir à la réussite. Ses capacités scientifiques et ses compétences étaient mon grand support.

La liberté qu'elle m'a accordée et les responsabilités qu'elle m'a confiées ont beaucoup contribué à ma formation et à mon autonomie de travail.

Je la remercie également pour m'avoir initié à la recherche et de m'avoir fait bénéficier de ses connaissances.

J'exprime ma gratitude et ma reconnaissance aux membres de jury, ainsi je les remercie chaleureusement d'avoir pris le soin et le temps de bien juger ce travail.

J'adresse également mes sincères remerciements à ma famille ainsi qu'à mes amis de m'avoir aidé à surmonter tous les obstacles et à me forger à travers les difficultés vécues durant toute cette période de travail.

A ceux qui ont été continuellement présents, qui m'ont épaulée par leur aide, soutien et encouragement.

Soyez assurés de mon profond respect.

** Monsieur Saal Aghiles **

Dédicaces

Je dédicace ce travail à :

- *A ma très chère mère Autant de phrases aussi expressives soient-elles ne sauraient montrer le degré d'amour et d'affection que j'éprouve pour toi. Tu m'as comblé avec ta tendresse et affection tout au long de mon parcours. Tu n'as cessé de me soutenir et de m'encourager durant toutes les années de mes études, tu as toujours été présente à mes cotés pour me consoler quand il fallait. En ce jour mémorable, pour moi ainsi que pour toi, reçoit ce travail en signe de ma vive reconnaissance et mon profond estime. Puisse le tout puissant te donner santé, bonheur et longue vie afin que je puisse te combler à mon tour.*

- *A Mon père ce travail te doit beaucoup, qu'il soit pour toi le témoignage de mon infinie reconnaissance pour ton soutien et ta compréhension durant les années de mes études.*

- *A ma très chère sœur Naima et à son mari Boussad pour leurs encouragements permanents, et leur soutien moral,*

- *A mes frères Omar et Amar pour leur appui et leur encouragement.*

- *Mes amis.*

Ce mémoire présente l'aboutissement du soutien et des encouragements qu'ils m'ont prodigués toute au long de ma scolarité, qu'ils ont soit remercier par cette modeste dédicace.

Aghiles

SOMMAIRE

Chapitre III : Analyse et conception

Chapitre IV : Réalisation

Liste des tables

Liste des figures

Notation

Description des symboles utilisés dans ce mémoire

$\preccurlyeq$: Relation d'ordre.

$A \backslash B$: Différence ensembliste.

$\cup$: Union.

$\cap$: Intersection.

$\subseteq$: Inclusion.

$\subset$: Inclusion sans égalité.

$|E|$: Cardinalité de l'ensemble E.

2^E : Ensemble des parties de E.

$\rightarrow$: Implication.

$\Leftrightarrow$: Equivalence.

Introduction générale

Introduction générale

La théorie de l'analyse de concepts formels (abrév. ACF) a été introduite par Wille en 1982 [Wille 1982]. Cette théorie permet d'induire des structures hiérarchiques de concepts formels à partir de structures (représentations) relationnelles. Elle a été utilisée dans divers domaines: psychologie, sociologie, médecine, biologie, linguistique, informatique, etc. [Wolff 1993].

L'analyse de concepts formels consiste à découvrir des clusters de connaissance à partir de relations binaires. Généralement, ces relations sont représentées sous forme de tables avec en ligne des objets et en colonne des attributs. Etant donné une relation R, l'intersection d'une ligne et d'une colonne (i.e. cellule) indique si un objet "*x*" vérifie ou ne vérifie pas l'attribut "*a*". Classiquement, les relations considérées sont non seulement binaires et booléennes mais aussi complètement renseignées (c.à.d. il est toujours connu si un objet possède ou ne possède pas un attribut).

Il s'avère que dans la réalité, certaines relations entre objets et attributs peuvent être partiellement connues, incertaines, imprécises, vagues, floues ou carrément inconnues. De ce fait, l'ACF est amenée à considérer des relations de diverses natures, modélisant des réalités concrètes (e.g. mesures, observations, jugements, etc.) où peuvent apparaître des questions de gradualité et/ou d'incertitude.

Dans [Obiedkov 2002] et [P.Burmeister and R.Holzer 2005], les auteurs ont soulevé la question de distinguer entre le cas où l'on sait qu'un objet ne possède pas un attribut et le cas où on ne sait pas si l'objet possède l'attribut ou non, c'est une distinction que les contextes habituels ne peuvent pas faire. Ils ont proposé d'introduire une troisième valeur, notée " ?", dans un contexte formel, ce qui conduit à la notion de contexte incomplet.

Dans [Ait-Yakoub et all, 2017], les auteurs se sont intéressés aux implications qui peuvent être généré à partir de ces contextes formels incomplets. À cette fin, ils ont défini ce

que l'on appelle "implications certaines" qui tiennent dans tous les mondes possibles compatibles avec l'information incomplète et les "implications possibles" qui tiennent dans au moins une situation compatible avec l'information incomplète. Une caractérisation des implications d'attributs possibles et certaines est aussi proposée à la fois pour la sémantique conjonctive et disjonctive.

Nous nous situons dans le contexte des représentations incomplètes, et nous nous intéressons particulièrement aux implications d'attributs possibles qui peuvent être généré à partir d'un contexte formel incomplet. Le travail qui nous a été proposé dans le cadre de ce mémoire est dans un premier temps d'implémenter une méthode qui permet de vérifier si une implication d'attributs donnée par un utilisateur est possible ou pas. Dans un deuxième temps de mettre en œuvre une approche qui permet de générer toutes les implications d'attributs possible (une base complète d'implications possibles).

Pour cela, nous avons subdivisé ce rapport en plusieurs volets comme suit :

Outre ce volet introductif, le premier chapitre porte sur l'analyse de concepts formels. Dans ce chapitre nous commençons par une représentation intuitive de l'AFC, Nous donnons par la suit une représentation basée sur des fondements mathématiques ainsi que des notions de fermeture de connexion de Galois. Et en fin nous donnons quelques domaines d'application de l'AFC.

Dans le deuxième chapitre, nous présentons la théorie des ensembles incomplets introduite par [Djouadi 2009]. On y détaille plus particulièrement les notions de base nécessaire à la compréhension de cette théorie. Nous commençons par donner une définition détaillée des contextes incomplets par la suite nous donnons les théorèmes mathématiques que nous allons utilisés pour la génération et la vérification des implications d'attributs possibles avec des exemples.

Dans le troisième chapitre nous présentons notre contribution qui consistera en l'analyse et réalisation d'une solution informatique pour une meilleure prise en charge des contextes formels incomplets plus précisément des implications d'attributs conjonctives possibles pour

un contexte formel incomplet donné. Nous commençons par présenter l'architecture générale de l'application par suite les algorithmes utilisés pour traduire le théorème mathématique.

Le quatrième chapitre est réservé à la mise en œuvre de notre application. Nous présentons les différents outils que nous avons utilisés, les techniques d'implémentations utilisées ainsi que des captures d'écrans qui illustrent le fonctionnement de notre application.

Nous terminons notre rapport avec une conclusion et quelques perspectives.

Chapitre I

Analyse de concepts formels

I.1. Introduction :

L'analyse de concepts formels a été introduite par Wille en 1982 [Wille 1982]. L'ACF permet d'induire des clusters de connaissances et se présente comme une méthode d'apprentissage et de représentation de connaissances. Elle a été utilisée dans divers domaines : psychologie, sociologie, biologie, médecine, linguistique, mathématiques, informatique, etc.

Nous commençons notre étude par une présentation intuitive de l'ACF sans pour autant introduire de formalisme mathématique. Par la suite nous consacrerons une section complète pour la présentation théorique de l'ACF sur la base de fondements mathématiques (algébriques). Pour bien fixer les idées nous devons connaitre c'est quoi un concept formel.

D'un point de vue psychologique, un concept formel est une unité de pensée constituée de deux parties, la partie "extension" et la partie "intension". La partie extension recouvre tous les objets du concept formel et la partie intension comprend tous les attributs possédés par tous ces objets. En effet les objets et les attributs jouent un rôle proéminent ensemble avec les différentes relations comme: la hiérarchie entre les concepts (sur-concept, sous-concept) la relation d'implication entres attributs et la relation d'incidence entre les paires (objet/ attribut) (un objet possède un attribut). L'idée de Wille était d'illustrer dans un premier temps les paires (objet/ attribut) ainsi que la relation d'incidence selon une représentation mathématique appelé contexte formel, ce dernier est considéré comme un outil de description des situations élémentaires sous la forme: l'objet " x " possède l'attribut "a". Dans un deuxième temps, il s'agira de découvrir les ensembles maximaux des objets satisfaisant un certain ensemble d'attribut.

Exemple:

Soit un ensemble d'animaux {lion, rouge-gorge, aigle, lièvre, autruche} décrit par rapport à certains de leurs attributs {prédateur, vole, ovipare, mammifère}. Le contexte formel

noté C (autrement dit la relation binaire) est représenté sous forme d'une table avec en lignes les animaux (correspondant aux objets) et en colonnes les attributs, telle que si l'objet " x_i" vérifie (resp. ne vérifie pas) l'attribut "a_j" alors la cellule "c_{ij}" est marquée par une croix (resp. reste vide).

Attribut / Animal	prédateur	vole	ovipare	mammifère
Lion	×			×
rouge-gorge		×	×	
Aigle	×	×	×	
Autruche			×	
Lièvre				×

Table I.1: Représentation du contexte formel

I.2. Rappel mathématique :

Dans cette section, nous rappelons quelques définitions nécessaires à notre travail. Ces définitions concernent les ensembles ordonnés. Ces définitions sont extraites de [Pasquier, 2000].

I.2.1. Un ensemble :

Intuitivement, un ensemble est une collection. Nous désignerons en général les ensembles par des lettres majuscules: A, B, C, D… etc. Les éléments d'un ensemble sont désignés en général par des lettres minuscules : a, b, c, d… etc. Si a est un élément d'un ensemble E, on écrit a ∈ E et on lit « a appartient à E » ou « a est élément de E ». Pour exprimer que a n'est pas un élément de E, on écrit : a ∉ E et on lit « a n'appartient pas à E ». Nous admettons l'existence d'un ensemble noté ∅, appelé ensemble vide, qui ne contient

aucun élément. Un ensemble réduit à un seul élément a est noté {a}. Plus généralement, un ensemble qui ne contient que les éléments $x_1/x_2/.../x_n$ est noté $\{x_i,..., x_n\}$. Si E est un ensemble et P un attribut vrai pour certains éléments de E, l'ensemble des éléments de E qui vérifient l'attribut P est souvent noté : $\{x : x$ vérifie P$\}$.

I.2.2. Relation binaire :

Une relation binaire R sur un ensemble E est un attribut portant sur les couples d'éléments de E. On notera aRb le fait que l'attribut est vraie pour le couple (a, b) ∈ E × E.

- R est réflexive si pour tout $x \in$ E, on a xR x ;

- R est symétrique si pour tout x, y ∈ E, on a xRy ⇒ yR x.

- R est transitive si pour tout x, y, z ∈ E, (xRy et y Rz) ⇒ xRz.

I.2.3. Relation d'ordre :

Une relation binaire R sur E est une relation d'ordre si et seulement si elle est réflexive, antisymétrique et transitive. On dit alors que E est un ensemble ordonné (par R). Une relation d'ordre est souvent notée ≼.

I.3. Théorie de l'analyse de concepts formels :

L'ACF fournit un cadre théorique pour l'apprentissage de la hiérarchie de concepts formels. Cet apprentissage s'effectue à partir d'un contexte formel $\mathcal{K} = (\mathcal{O}, \mathcal{P}$, R) où R est une relation binaire complètement définie entre un ensemble d'objets $\mathcal{O}$ et un ensemble d'attributs $\mathcal{P}$. Généralement la relation R est représentée sous forme d'une table avec en lignes des objets et en colonnes des attributs telle que : la présence d'une croix (×) (resp. l'absence) dans la cellule c_{ij} indique que l'objet x satisfait (resp. ne satisfait pas) l'attribut. [Seddoud, 2011].

Soulignons qu'une croix peut être aussi représentée par la valeur 1.Dans la suite de ce mémoire nous utiliserons indifféremment les termes objets et attributs pour désigner les éléments des ensembles $\mathcal{O}$ et $\mathcal{P}$.

Dans ce qui va suivre on va définir les notions de bases de l'analyse de concepts formels à savoir le contexte formel binaire, le concept formel et l'opérateur de dérivation de Galois.

I.3.1. Contexte formel :

Un Contexte Formel est un triplet $\mathcal{K}$ = ($\mathcal{O}$, $\mathcal{P}$, R) où $\mathcal{O}$ est un ensemble d'objets, $\mathcal{P}$ est un ensemble d'attributs et R une relation binaire entre $\mathcal{O}$ et $\mathcal{P}$ vérifiant: R $\subseteq \mathcal{O}\times\mathcal{P}$; (o, p) $\in$ R avec o $\in \mathcal{O}$et p$\in \mathcal{P}$ signifie que l'objet o possède l'attribut p ou que l'attribut p est possédé par l'objet o. Un contexte formel peut être représenté sous la forme d'un tableau où les lignes correspondent aux objets et les colonnes correspondent aux attributs. Les cases du tableau sont remplies comme suit :

Si l'i-ème objet o est en relation R avec le j-ème attribut P alors la case intersection de la ligne i et la colonne j contient "1" sinon la case est vide.

Reprenons l'exemple de la Table I.1 : $\mathcal{O}$ est l'ensemble des objets qui sont {lion, rouge-gorge, aigle, Autruche, lièvre}. $\mathcal{P}$ est l'ensemble des attributs {prédateur, vole, ovipare, mammifère}. R représente les relations entre les objets et les attributs de ce contexte par exemple ({lièvre}, {mammifère}) ici les relations sont définis par des croix (×).

I.3.2. Concepts formels :

Un Concept Formel d'un contexte $\mathcal{K}$ = ($\mathcal{O}$, $\mathcal{P}$, R) est une paire (X, A) avec : X $\subseteq \mathcal{O}$, A $\subseteq \mathcal{P}$, X^{Δ} =A et A^{Δ} = X, où X^{Δ} est l'ensemble de tous les attributs de A possédés par les objets de X et de façon duale A^{Δ} est l'ensemble de tous les objets possédant les attributs de

X. Les ensembles X et A sont appelés respectivement extension et intension du concept formel C.

Pour expliquer la notion de concept formel on va reprendre la table I.1, nous considérons tous les attributs du rouge-gorge {vole, ovipare} et posons-nous la question: quels sont les animaux satisfaisant ces attributs ? Nous obtenons alors l'ensemble X = {aigle, rouge gorge}. Nous constatons que l'ensemble X est l'ensemble maximal des animaux satisfaisant tous les attributs de l'ensemble A = {vole, ovipare}.

Il résulte que X est l'ensemble de tous les animaux vérifiant tous les attributs de A et A est l'ensemble de tous les attributs vérifiés par tous les animaux de X. La paire (X, A) est appelé concept formel. Tandis que X est appelé extension et A est appelé intension.

I.3.3. Hiérarchie entre concepts formels :

Il y a une relation d'ordre partiel c.à.d. la relation "Sous-concept, Sur-concept".

Etant donné deux concepts formels (X, A), et (Y, B). On dit que (X, A) est un sous-concept de (Y, B), (dualement (Y, B) est un sur-concept de (X, A)) si l'extension de (X, A) est un sous ensemble de l'extension de (Y, B). c.à.d. $X \subseteq Y$ (dualement: l'intension de (X, A) est un sur-ensemble de l'intension de (Y, B). c.à.d. $A \supseteq B$).

Reprenons l'exemple précédent. Etant donné les deux concepts formels suivants:

({aigle}, {prédateur, vole, ovipare}) et ({aigle, rouge-gorge}, {vole, ovipare}). ({aigle}, {prédateur, vole, ovipare}) est un sous-concept de ({aigle, rouge-gorge}, {vole, ovipare}).

Dualement, ({aigle, rouge-gorge}, {vole, ovipare}) est un sur concept de ({aigle}, {prédateur, vole, ovipare}).

L'extension {Aigle} du sous-concept est un sous-ensemble de l'extension {aigle, rouge-gorge} du sur concept. De la même manière l'intension {prédateur, vole, ovipare} du sous-concept est un sur-ensemble de l'intension {vole, ovipare} du sur-concept.

Les concepts formels sont induits en utilisant l'operateur Ensembliste (.) $^{\Delta}$ (appelé operateur de dérivation de Galois).

I.3.4. Operateur de dérivation de Galois :

Wille propose l'utilisation d'un opérateur de dérivation de Galois. Cet opérateur est différemment noté dans la littérature par : (.) $*$, (.) $\uparrow$, (.) $\downarrow$. Dans notre mémoire, nous appellerons l'opérateur classique de Wille: opérateur Ensembliste (.) $^{\Delta}$. [Wille 1982]

Soit K un contexte formel de la figure Table I.1 $\mathcal{K} = (\mathcal{O}, \mathcal{P}, R)$ est un triplet, ou R est une relation binaire entre certains objets de $\mathcal{O}$ et certains attributs de $\mathcal{P}$.

X^{Δ} correspond aux attributs qui sont satisfaits par tous les objets de X: $X^{\Delta} = \{a \in \mathcal{P} \mid \forall\, x \in \mathcal{O}\ (x \in X \Rightarrow (x, a) \in R)\}$

{Lion, aigle} $^{\Delta}$ = {Prédateur}

A^{Δ} correspond aux objets qui satisfont tous les attributs de A:
$A^{\Delta} = \{x \in \mathcal{O} \mid \forall\, a \in \mathcal{P}\ (a \in A \Rightarrow (x, a) \in R)\}$

{Mammifère} $^{\Delta}$ = {lion, lièvre}

Une paire ({X, A}) tel que $X^{\Delta} = A$ et $A^{\Delta} = X$ est appelée concept formel.
({lion, lièvre} {mammifère}) est un concept formel.
{lion, lièvre} est appelé extension, {mammifère} est appelé intension ($\preccurlyeq$) définit une relation d'ordre entre deux concepts formels: <X1, A1> $\preccurlyeq$ <X2, A2> ssi $X1 \subseteq X2$ (ou $A2 \subseteq A1$)

({lion}, {Prédateur, Mammifère}) $\preccurlyeq$ ({lion, aigle} {Mammifère}) .

I.4. Implication d'attributs dans un contexte formel :

I.4.1. Règles d'associations :

Introduite par Agrawal. L'extraction de règles d'association est l'un des principaux problèmes du processus d'extraction de connaissance dans les bases de données.

L'extraction des règles d'association a pour but d'identifier des associations significatives entre les données. Les relations ainsi identifiées peuvent être utiles pour de nombreux organismes commerciaux, scientifiques, industriels et de gestion de l'information, afin d'améliorer leurs objectifs dans leurs activités. Par exemple, il est important pour toute entreprise commerciale de recueillir des informations variées sur le comportement de ses clients (préférences, habitudes) afin de mener diverses analyses et d'en déduire des modalités d'actions. A ce titre, l'analyse des factures (tickets de caisse) d'un supermarché permettra d'étudier les habitudes des clients (citons par exemple : les clients qui achètent du lait ont tendance à acheter aussi du café et du sucre.), en fonction de quoi il sera possible de réorganiser les rayons du magasin afin d'améliorer les ventes.

L'exemple le plus populaire est celui du « panier de la ménagère » qui consiste à enregistrer l'ensemble des articles présents dans le panier d'un consommateur. Dans cet exemple, chaque client est un objet et les articles achetés sont ses attributs, c'est pour quoi on parle très souvent d'item pour désigner un attribut. Pour cela nous utiliserons dans la suite de cette sous-section indifféremment les termes items et attributs pour désigner les propriétés. Les premiers algorithmes ont fonctionné sur de tels exemples afin de découvrir les associations cachées entre les articles achetés ensemble. En extrayant des regroupements des articles achetés par une même personne, nous pouvons prendre plusieurs décisions.

Généralement une règle d'association entre un sous-ensemble d'items (attributs) A un sous-ensemble d'items (attributs) B est notée : $A \rightarrow B$. Un processus de découverte des règles d'association consiste à extraire toutes les règles d'association telles que :

Support (A→B) ≥ min-support et confiance (A→B) ≥ min-confiance où support et confiance sont définis comme suit :

- Support(A) = $\frac{|A^{\Delta}|}{|\mathcal{O}|}$
- Support (A→B) = $\frac{|(A \cup B)|^{\Delta}}{|\mathcal{O}|} = \frac{|A^{\Delta} \cap B^{\Delta}|}{|\mathcal{O}|}$
- Confiance(A→B) = $\frac{\text{support}(A \rightarrow B)}{\text{support}(A).}$

min-support et min-confiance sont des paramètres fixés à priori. D'autre part, un sous ensemble d'items est dit fréquent si support ( ) ≥ min-support. A ce titre, la première approche proposée [Agrawal 1994] pour la découverte des règles d'association a consisté à découvrir les ensembles d'items fréquents.

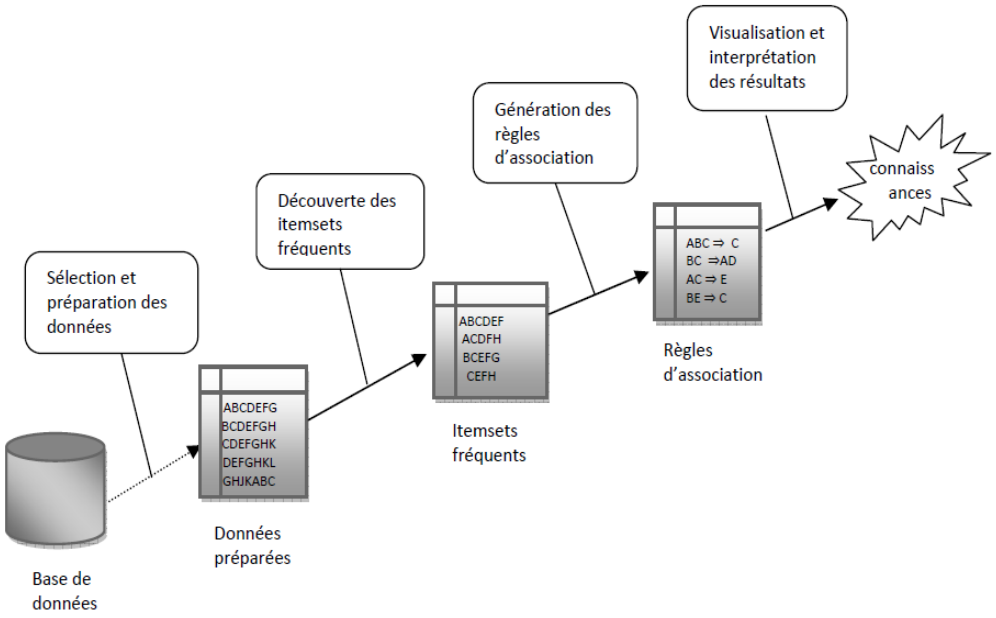


Figure 1.1 : Etapes du processus d'extraction des règles d'association.

I. 4.2. Implications d'attributs conjonctives dans un contexte formel binaire :

Soient un contexte formel $\mathcal{K} = (\mathcal{O}, \mathcal{P}, R)$ et B1;B2 $\in \mathcal{P}$ deux ensembles d'attributs. On dit que B1 implique B2 si et seulement tout objet de $\mathcal{O}$ qui a les attributs de B1 a aussi les attributs de B2. $B1 \rightarrow B2$ ssi $B1^{\Delta} \subseteq B2^{\Delta}$

Dans le contexte formel des animaux donné dans la Table I.1 on a l'exemple de l'implication suivante:

Vole → Ovipare, avec $B1^{\Delta}$ = {rouge-gorge, aigle} ⊆ $B2^{\Delta}$ = {rouge-gorge, aigle, Autruche}

I.5. Domaine d'application :

L'analyse de concepts formels donne lieu actuellement à diverses applications notamment dans l'aide à la décision, la découverte de connaissances, la recherche d'informations, etc.

I.5.1 Recherche d'informations :

La recherche d'information est une branche de l'informatique qui s'intéresse à l'acquisition, l'organisation, le stockage, la recherche et la sélection d'information. Pour une vision simple, la recherche d'information consiste à "trouver les documents correspondent aux mots de la requête posée par l'utilisateur ". Pour ce faire, on est amené à considérer une relation binaire Documents×Termes. Cette relation est souvent représentée par une table avec en ligne des documents et en colonne des termes. Etant donné une relation R, l'intersection d'une ligne et d'une colonne indique si un terme « t » appartient ou n'appartient pas au document « d ».

Exemple

Etant donné un ensemble fini de documents D, un ensemble fini de termes T et une relation binaire R telle que pour un couple $(t_i, d_j) \in R$ représente le fait que le terme $t_i \in T$ est présent dans le document $d_j \in D$.

$\mathcal{R}$	t1	t2	t3	t4
d1	×		×	
d2				×
d3		×	×	
d4	×	×	×	×

Table I.2: Relation binaire terme-document

L'évidente analogue entre la relation binaire Objets×Attributs caractérisant l'ACF et une relation d'incidence de type Documents×Termes caractérisant la recherche d'information à rapidement suscité un engouement certain pour l'utilisation de l'ACF en recherche d'information. Dans ce cas les documents correspondent à des objets formels et les termes d'indexation (descripteurs, éléments de thésaurus, etc.) correspondent aux propriétés formelles [Priss 2000]. Les concepts formels résultant d'une telle relation peuvent être interprétés comme des paires ({réponse}, {requête}) où la requête correspond à l'intension du concept tandis que la réponse correspond à son extension. La relation de subsomption (relation d'ordre partiel) entre concepts formels peut être considérer comme une relation de spécialisation/généralisation entre requêtes.

Nous pouvons remarquer que dans le contexte formel illustré dans la table I.2, les seules informations que nous pouvons extraire c'est l'appartenance ou non appartenance du terme au

document. Cependant dans le domaine de la recherche d'informations cette appartenance n'est pas toujours égale à 0 ou à1, ma.is appartient plutôt à l'intervalle [0, 1].

I. 5.2. La classification de données par le treillis de concepts :

La classification est une des tâches centrales de l'étape de fouille de données dans le processus d'extraction de connaissances dans les bases de données. Le problème de la classification est traité dans plusieurs communautés de recherche qui se découvrent et s'enrichissent mutuellement : statistiques, apprentissage automatique, réseaux de neurones et raisonnement à partir de cas.

Les treillis de concepts formels (ou treillis de Galois) sont une structure mathématique permettant de représenter les classes non disjointes sous-jacentes à un ensemble d'objets (exemples, instances, tuples ou observations) décrits à partir d'un ensemble d'attributs (propriétés, attributs, descripteurs ou items). Ces classes non disjointes sont aussi appelées concepts formels, hyper-rectangles ou ensembles fermés. Une classe matérialise un concept (à savoir une idée générale que l'on a d'un objet). CHARADE (Cubes de Hilbert Appliqués à la Représentation et à l'Apprentissage de Descriptions à partir d'Exemples) est la première méthode à utiliser la correspondance de Galois, pour générer un ensemble de règles d'association non redondantes dont le but peut être restreint à une classe prédéfinie.

Pour limiter l'espace de recherche, plusieurs heuristiques sont définies comme des paramètres que l'utilisateur peut sélectionner en fonction des propriétés des règles qu'il souhaite obtenir ou de la nature du problème qu'il traite. Le principe de classement à partir de cet ensemble de règles n'est pas explicitement formulé par l'auteur.

L'utilisateur peut ensuite les employer pour construire sa méthode de classement [Ganascia 1993] mentionne un certain nombre d'extensions de CHARADE ayant permis de traiter des applications de grande taille, notamment pour la catégorisation de textes, la découverte médicale, l'apprentissage de caractères chinois et l'extraction de connaissances dans une base de données épidémiologiques. Plusieurs méthodes de classification basées sur

le treillis de concepts ont été développées et comparées à des méthodes standards couramment utilisées pour ce type de problèmes.

Exemple

La table I.1: Représentation du contexte formel représenté avec $\mathcal{K} = (\mathcal{O}, \mathcal{P}, R)$ avec

$\mathcal{O}$ = {Lion, rouge-gorge, Aigle, Autruche, Lièvre} et $\mathcal{P}$ = {prédateur, vole, ovipare, mammifère}.

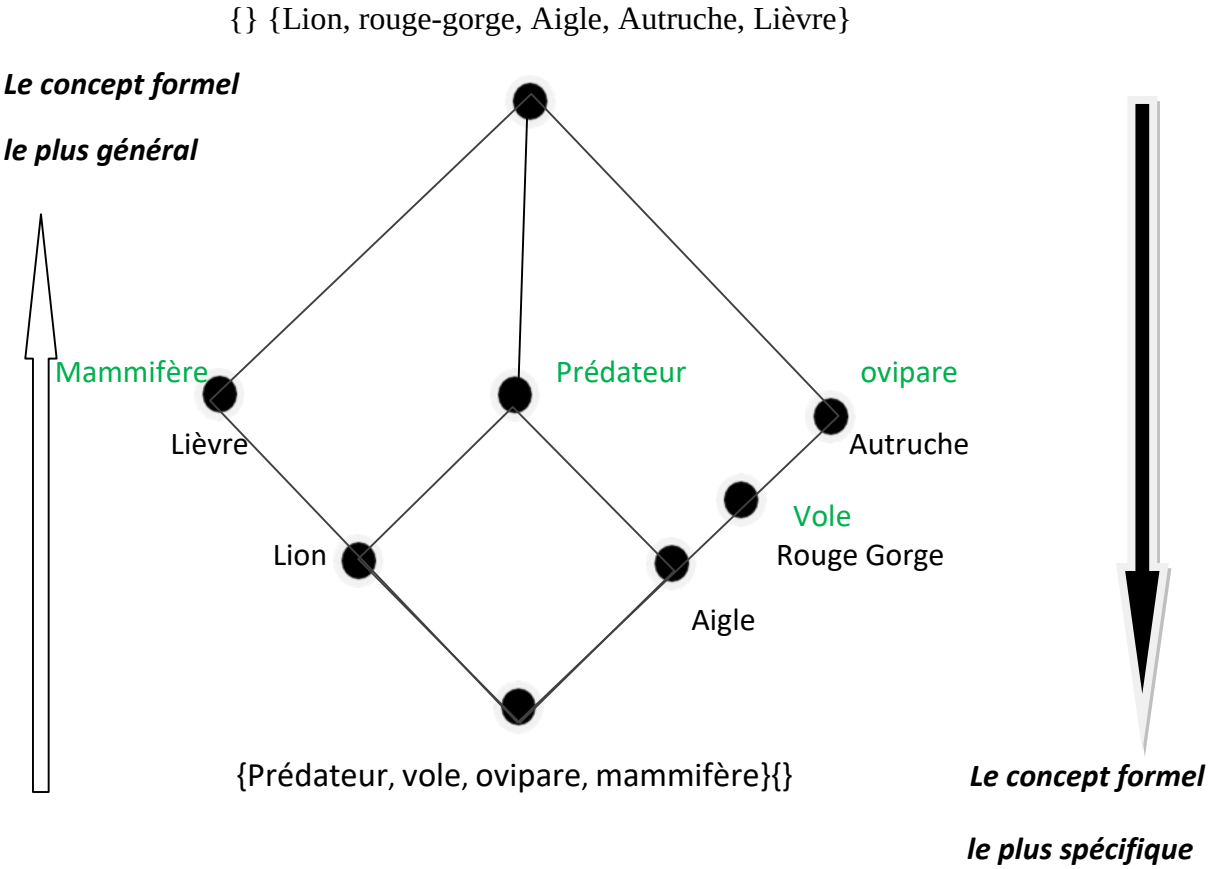


Figure I.2 : Hiérarchie (treillis) de concepts formels

✓ **Explication de la figure I.2**

Cette figure est constituée de nœuds et de segments. Elle comporte aussi les noms de tous les objets et tous les attributs du contexte. Chaque nœud correspond à un concept formel. La Figure peut être lue comme suit: A chaque fois qu'un nœud "*n*" est étiqueté par un attribut "a", tous les objets descendants de ce nœud "*n*" héritent l'attribut "*a*". De façons duale, à chaque fois qu'un nœud "*n*" est étiqueté par un objet " x ", " x " est hérité vers le

haut et tous les ancêtres de nœud "*n*" le partage. Ainsi l'extension "X" d'un concept **<X, A>** correspondant au nœud "*n*" est obtenue en considérant tous les objets qui apparaissent sur les descendants du nœud "*n*" dans le treillis et son intension A est obtenue en considérant tous les attributs qui apparaissent sur les ancêtres du nœud "*n*" dans le treillis.

Nous pouvons remarquer que dans cet exemple, l'intension du concept formel sommet correspond à l'ensemble vide, tandis que l'extension de concept formel sommet correspond à l'ensemble de tous les animaux. Nous pouvons trouver toute fois un concept formel sommet différent de l'ensemble vide.

I.6. Conclusion :

Dans ce chapitre nous avons d'abord présenté l'analyse de concepts formels de manière intuitive afin d'en faciliter la compréhension. Nous avons rappelé par la suite les notions mathématiques sous-jacentes à la théorie de l'analyse de concepts formels. Nous avons essentiellement mis l'accent sur certaines propriétés théoriques de l'analyse de concepts formels. Nous avons présenté les implications d'attributs dans un contexte formel .Enfin nous avons montré quelques applications de l'analyse de concepts formels.

Il s'avère que l'analyse de concepts formels basée sur des relations Booléennes (ACF classique) peut présenter certaines insuffisances car la réalité présente souvent des valeurs qui ne sont pas totalement renseignées ou certaines mais de diverses natures (floues, incomplètes,…).

Pour cela, nous sommes amenés à considérer des cadres théoriques plus appropriés pour étendre l'analyse de contextes formels classique à l'analyse de contextes formels incomplets. A ce titre, le chapitre suivant présente la théorie des ensembles.

Chapitre II

Implications d'attributs dans un contexte formel incomplet

II.1. Introduction :

Dans le modèle de base proposé par Wille [Wille, 1982], l'analyse de concepts formels concerne des relations binaires et booléennes appelées contextes formels. Ce modèle repose sur les deux suppositions suivantes :

- La relation est Booléenne et ne peut prendre une autre valeur que vrai ou faux.
- L'information est complète (parfaitement renseignée) c.à.d. il est toujours connu que l'objet possède ou ne possède pas l'attribut).

Il s'avère que l'analyse de concepts formels est souvent amenée à considérer des contextes formels (relations) de diverses natures, modélisant des réalités concrètes (mesure, observation, jugement, etc.). Où peuvent apparaitre des données incomplètes (imprécises, incertaines, floues, vagues, partiellement renseignées, ou même manquantes).[Djouadi, 2009]. Dans pareils cas, le modèle de base proposé par Wille s'avère inapproprié. L'exemple de la table II.1 illustre différents cas de données incomplètes.

Attributs / Objets	Jeune	Anglais	Marié(e)
Farid	0.6	[0.2, 0.4]	Oui
Amar	1	0	(0.6 ; 0.0)
Kenza	0.8	]0, 1]	Non
Karim	0.7	0.8	?

Table II.1 : Contexte formel flou

La première colonne de la table **II.1** contient des valeurs appartenant à l'intervalle [0, 1]. Elle indique le degré de satisfaction de la propriété graduelle *Jeune.* La seconde colonne indique le niveau de maîtrise de l'*Anglais*. Ce niveau peut être connu de manière précise ou seulement apprécié sous forme d'intervalles. La troisième colonne illustre la présence d'incomplétude (cas de Karim) et d'incertitude (cas de Amar) : Alors que Farid est *marié* et

Kenza ne l'est pas, rien ne peut être affirmé pour Karim,' ?' qui signifie l'absence totale de valeur ce qui correspond à (0 ; 0).Dans cette optique, Burmeister et Holze [P.Burmeister and R.Holzer 2005] ont considéré une troisième valeur qui permet de modéliser l'ignorance totale sur la satisfaction de l'attribut par l'objet.

En modélisant l'état épistémologique d'une connaissance partielle (cas partiellement informé) sur le mariage de Amina par la théorie des possibilités [Zadeh 1978], la paire (0.7 ; 0.0) exprime que *la possibilité* que Amina ne soit pas *marié* est de 0.0 et de 0.7 qu'elle le soit.

Comme on l'a précisé, L'AFC est amenée a considérer des contextes de divers natures, il semble donc utile de voir ce que deviennent les implications résultant de chaque contexte formel. Dans ce chapitre, nous allons traiter le cas des contextes formels incomplets qui correspond à la troisième colonne (ignorance totale sur la satisfaction de la propriété par l'objet). De manière originale, nous allons définir et caractériser de nouvelles formes d'implications d'attributs, a savoir les implications et certaines que ce soir dans le cas conjonctif ou dans le cas disjonctif. Le cadre théorique retenu pour modéliser l'incomplétude est celui de la théorie des possibilités.

II.2. Contexte incomplet :

Il est largement admis que dans de nombreux domaines, les connaissances peuvent être incomplètes. Il semble donc important de distinguer entre le cas ou l'on sait qu'un objet ne possède pas un attribut et le cas ou on ne sait pas si un objet possède l'attribut ou non. C'est une distinction que les contextes formels classiques ne peuvent pas faire. Il est donc utile de voir ce que deviennent les implications résultant d'un contexte formel incomplet. Le cas de l'ignorance partielles dans des contextes a été considéré par Obiedkov [Obiedkov 2002] en utilisant la logique modale et par Burmeister et Holzer en utilisant la logique a trois valeurs (Kleen three-valued logic). Ils ont proposé d'introduire une troisième valeur, notée " ? ", Dans un contexte formel, ce qui conduit à la notion de contexte incomplet, parfois aussi appelé contexte formel à trois valeurs.

Dans le cadre d'un contexte incomplet, la présence des valeurs manquantes cause un désagrément lors de l'extraction de connaissances. Par exemple, les valeurs manquantes posent un problème lors de l'évaluation du support d'un itemset. Par conséquent, cette évaluation doit être aménagée en fonction de ces valeurs manquantes. Ainsi, plusieurs solutions ont été proposées dans ce cadre. Kryszkiewicz propose de considérer deux stratégies lors de l'évaluation du support. Une stratégie dite optimiste, pour laquelle l'itemset est supposé présent. Ceci donne la valeur optimiste du support. En outre, une stratégie dite pessimiste, pour laquelle l'itemset est supposé absent, permet de donner la valeur pessimiste du support. Ces solutions s'avèrent inadaptées dans le cas d'une base contenant plusieurs valeurs manquantes. Une autre solution proposée par le même auteur, consiste en la redéfinition de la notion de support pour l'adapter à la présence de valeurs manquantes. Dans ce qui suit, nous présentons brièvement cette solution.

Dans ce qui suit, nous utilisons la théorie des possibilités pour modéliser des informations incomplètes.

Un contexte incomplet est formellement représenté par $\mathcal{K}^{?}$ =($\mathcal{O}$, $\mathcal{P}$,{+,-, ?}, R) ou $\mathcal{O}$ est l'ensemble d'objets , $\mathcal{P}$ un ensemble d'attributs ,"+","-"," ?" sont les trois entrées possibles du contexte incomplet ,et R est une relation ternaire R $\subseteq \mathcal{O}\times\mathcal{P} \times${+,-, ?}.

L'interprétation de la relation R est comme suit. Doit $\in \mathcal{O}$ et a $\in \mathcal{P}$:

- (x, a, +) $\in$ R : il est connu que l'objet x vérifie l'attribut a.
- (x, a, -) $\in$ R : il est connu que l'objet x ne vérifie pas l'attribut a.
- (x, a, ?) $\in$ R : on ne sait pas si l'objet x vérifie l'attribut a ou pas.

Exemple

$\mathcal{K}^{?}$	a1	a2	a3
x 1	+	+	-
x 2	-	?	+
x 3	?	+	-

Table II.2 : Contexte formel incomplet

Un contexte formel incomplet peut être considéré comme la famille de tous les contexte formels standards obtenus en remplaçant les entrés inconnues (x, a, ?) par celles connues ((x, a,-) ou (x, a,+)). Considérant les deux cas extrêmes ou toutes les entrées inconnues les entrés (x,a, ?) sont remplacées par (x, a,-) , et le cas ou elles sont remplacées par (x, a,+), cela donne naissance aux complétions inférieures et supérieures respectivement.

De cette manière, deux contextes formels classiques (booléens), notés k^* et K_* peuvent être obtenus respectivement suite aux deux remplacements. De manière formelle :

- $\mathcal{K}_* = (\mathcal{O}, \mathcal{P}, R_*)$ est un contexte formel booléen tel que $R_* = \{(x, a)|(x, a,-) \in R\}$
  Ou les entrées " ?" sont remplacées par -, interprétant le manque de connaissance sur (x, a) comme le fait que x ne possède pas l'attribut a.

- $\mathcal{K}^* = (\mathcal{O}, \mathcal{P}, R^*)$ est un contexte formel booléen tel que $R^* = \{(x, a)|(x, a,+) \in R\}$ ou les entrées " ?" sont remplacées par +, interprétant le manque de connaissance sur (x, a) comme le fait que x possède l'attribut a.

Il existe exactement 2^n contextes formels obtenus en remplaçant chaque " ?" par "+"ou par "-"(avec n le nombre de " ?" dans le contexte formel incomplet). Ils sont appelés contextes possibles. Si $\mathcal{K}_1$ et $\mathcal{K}_2$ sont des contextes formel obtenus a partir de $\mathcal{K}^?$. On définit un ordre entre eux comme suit : $\mathcal{K}_1 < \mathcal{K}_2$ si $R_1 \subset R_2$ i.e., il y a plus de "+" dans R_2 que dans R_1. Il est facile de vérifier que :

Lemme1. $\mathcal{K}_1 \prec \mathcal{K}_2$ implique que pour tout sous-ensemble A d'attributs $A^{\Delta}_{K1} \subseteq A^{\Delta}_{K2}$.

Preuve du Lemme1 : Il est évident que puisque tous les "+" dans R_1 sont des "+" dans R_2 il ne peut pas y avoir moins d'objets satisfaisant les attributs dans A pour $\mathcal{K}_2$ que pour $\mathcal{K}_1$.

Il est clair que le contexte minimal est $\mathcal{K}_*$ et le contexte maximal est $\mathcal{K}^*$.

Exemple

Dans les tableaux on va illustrer tous les contextes formels possibles obtenus à partir de la Table **II.2** proposé dans l'exemple précédent.

K^*	a1	a2	a3
x1	+	+	-
x2	-	+	+
x3	+	+	-

K_*	a1	a2	a3
x1	+	+	-
x2	-	-	+
x3	-	+	-

K_1	a1	a2	a3
x1	+	+	-
x2	-	+	+
x3	-	+	-

K2	a1	a2	a3
x1	+	+	-
x2	-	-	+
x3	+	+	-

Tables II.3 : Tous les contextes formels possibles

Dans le cas d'un contexte formel incomplet $\mathcal{K}^?$, l'implication d'attribut a1 →a2 est valide dans le contexte formel $\mathcal{K}^*$, $\mathcal{K}_*$, $\mathcal{K}_1$ et $\mathcal{K}_2$; cette implication d'attributs est valide indépendamment de ce que peuvent représenter les points d'interrogation. Considèrent l'implication d'attribut a3 → a2, elle n'est valable que dans $\mathcal{K}^*$ et $\mathcal{K}_1$: elle dépend de la valeur avec laquelle les points d'interrogation seront remplacés.

Dans ce qui va suivre nous allons utiliser la terminologie A^{Δ}_{K-} pour designer $\{x \mid \subseteq R_*(x)\}$ l'ensemble des objets possédant certainement tous les attributs de A tandis que A^{Δ}_{k+} pour désigner $\{x \mid \subseteq R^*(x)\}$ est l'ensemble des objets pouvant posséder tous les attributs de A.

II.3. Implication d'attributs conjonctives dans un contexte formel incomplet :

D'ordinaire, l'implication $p \rightarrow q$ exprime une liaison entre les valeurs des propositions *p* et *q*. Elle rend vraie lorsque la proposition *q* est vraie ou quand la proposition *p* est fausse, d'où les valeurs de vérité données dans la table II.2.

p \ *q*	**Vrai**	**Faux**
Vrai	Vrai	Faux
Faux	Vrai	Vrai

Table II.4: Table de vérité de l'implication classique $p \rightarrow q$

Les implications d'attributs (conjonctives) obtenues à partir du contexte formel incomplet sont soit des implications d'attributs certaines, soit des implications d'attributs possibles.

II.3.1. Implications d'attributs certaines :

Une implication d'attributs et dite certaine dans un contexte incomplet $\mathcal{K}^?$ si et seulement si elle est valide dans chaque contexte formel $\mathcal{K}$ tel que $\mathcal{K}_- \preccurlyeq \mathcal{K} \preccurlyeq \mathcal{K}^+$.

Cette définition est difficile à vérifier en utilisant une énumération explicite des contextes formels possible. C'est pour cela qu'on fait recours au théorème suivant qui fournit une solution à ce problème.

Théorème 1 [Ait Yakoub and All, 2017]

A→B est une implication d'attributs certaine dans $\mathcal{K}^?$ si et seulement si $A^{\Delta}_{K+} \subseteq A^{\Delta}_{K-}$

Preuve du théorème :

a) condition nécessaire

Soit A→B une implication d'attributs certaine dans $\mathcal{K}^?$ et on suppose que $A^{\Delta}_{K+} \nsubseteq A^{\Delta}_{K-}$

$A^{\Delta}_{K+} \nsubseteq A^{\Delta}_{K-} \Rightarrow \exists\, x \in \mathcal{O} \mid x \in A^{\Delta}_{K+}$ et $x \notin A^{\Delta}_{K-} \Rightarrow \exists$ un contexte formel $\mathcal{K}_j (\mathcal{O}, \mathcal{P}, R_j)$ tel que $R_j = R \cup Q$ ou $Q = \{(x, a) \in R^{+} | a \in A \cap (x, a, ?) \in R\}$

Ainsi, $x \in A^{\Delta}_{kj} \Rightarrow x \notin B^{\Delta}_{kj} \Rightarrow A^{\Delta}_{kj} \nsubseteq B^{\Delta}_{kj} \Rightarrow$ A→B n'est pas valide dans $\mathcal{K}_j \Rightarrow$ A→B n'est pas une implication d'attributs certaines.

Une autre preuve directe peut être donnée :

Notons A→B est une implication d'attribut certaine si et seulement si a→b est une implication d'attribut certaine pour tout b∈ B\A. Dans ce qui suit nous pouvons limiter aux implications d'attributs certaines A→b. Cela signifie que pour tout contexte $\mathcal{K}$ compatible avec $\mathcal{K}^?$, $A^{\Delta}_{K} \subseteq (b)^{\Delta}_{K}$ est valide.

Considérons l'ensemble des objets $\mathcal{O}_A = \{ x \in \mathcal{O} : \forall a \in A, (x, a, -) \notin R \}$. Ainsi, les lignes restreintes à A du contexte incomplet correspondant à O_A ne contiennent que "+"ou" ?" .L'inclusion $A^{\Delta}_{K} \subseteq (b)^{\Delta}_{K}$ $\forall\, \mathcal{K}$ compatible avec $\mathcal{K}^?$ signifie que si $x \in \mathcal{O}_A$ alors $(x, b, +) \in R$, donc nous pouvons voir que $A^{\Delta}_{K+} = \mathcal{O}_A$ tandis que $(b)^{\Delta}_{K-} \supseteq \mathcal{O}_A$.

b) Condition suffisante

Si $A^{\Delta}_{K+} \subseteq A^{\Delta}_{K-}$ alors A→B est une implication d'attribut certaine dans $\mathcal{K}^?$. En effet pour tout contexte formel possible $\mathcal{K}_j$ il est valide que $A^{\Delta}_{Kj} \in A^{\Delta}_{K+} \subseteq A^{\Delta}_{K-} \in B^{\Delta}_{Kj}$ due au Lemme-1 par conséquence A→B est une implication d'attributs certaine dans $\mathcal{K}_j$ pour tous les contextes formels possibles $\mathcal{K}^?$.

II.3.2. Implications d'attributs possibles :

Une implication d'attribut est dite possible dans un contexte incomplet $\mathcal{K}^?$ si et seulement si elle est valide dans au moins un contexte formel $\mathcal{K}$ tel que $\mathcal{K}_- \preccurlyeq \mathcal{K} \preccurlyeq \mathcal{K}^+$.

Il est clair qu'une implication d'attributs certaine est une implication d'attribut possible dans le sens de cette définition et l'inverse ne tient pas .Vérifier si une implication d'attributs est possible exige l'énumération de tous les contextes possibles. Mais avec le théorème suivant on facilite cette énumération.

Théorème 2 [Ait Yakoub and All, 2017]

A→B est une implication d'attributs possible dans $\mathcal{K}^?$ si et seulement si $A^{\Delta}_{K-} \subseteq B^{\Delta}_{K+}$

Preuve du théorème :

a) condition nécessaire

A→B est une implication d'attributs possibles dans $\mathcal{K}^?$ signifie qu'il existe un contexte formel $\mathcal{K}_j$ tel que $\mathcal{K}_- \preccurlyeq \mathcal{K}_j \preccurlyeq \mathcal{K}^+$ et $A^{\Delta}_{Kj} \subseteq B^{\Delta}_{K+}$.

Il est clair que $A^{\Delta}_{K-} \subseteq A^{\Delta}_{Kj}$ et $B^{\Delta}_{Kj} \subseteq B^{\Delta}_{K+}$ par le Lemme 1.

Mettre ces inclusion ensemble : que $A^{\Delta}_{K-} \subseteq A^{\Delta}_{Kj} \subseteq B^{\Delta}_{Kj} \subseteq B^{\Delta}_{K+}$, enfin $A^{\Delta}_{K-} \subseteq B^{\Delta}_{K+}$.

b) condition suffisante

On suppose que $A^{\Delta}_{K-} \subseteq B^{\Delta}_{K+}$. Deux cas sont facilement donnés avec :

B⊆A ≠∅ et B⊈A alors : A→B est une implication valide si et seulement si A→B\A est une implication d'attribut valide dans le même contexte. Donc nous n'avons qu'à étudier le cas ou A et B sont disjoints. Définir le contexte formel $\mathcal{K}_{AB}$ comme suit : Remplacer tout " ?" dans les colonnes de A par "-" et tout les " ?" dans les colonnes B par des "+". Alors, il est clair que $A^{\Delta}_{K} = A^{\Delta}_{KAB}$ et $B^{\Delta}_{K+} = B^{\Delta}_{KAB}$ .Par conséquent $A^{\Delta}_{KAB} \subseteq B^{\Delta}_{KAB}$ donc A→B est une implication d'attributs valide dans K_{AB}, et puisque $\mathcal{K}_- \preccurlyeq \mathcal{K}_{AB} \preccurlyeq \mathcal{K}^+$, A→B est une implication d'attributs possible dans $\mathcal{K}^?$.

II.4. Implications d'attributs disjonctives possible et certaines dans un contexte incomplet :

Comme dans le cas des implications d'attributs conjonctives, nous distinguons les implications d'attributs disjonctives certaines et les implications d'attributs disjonctives possibles.

Notons ($A)^{\prod}_{K-}$ est l'ensemble des objets ayant certainement au moins un attribut dans A et ($A) =^{\prod}_{K+}$) est l'ensemble des objets pouvant avoir au moins un attribut dans A. Alors $\overline{(A)^{\prod}_{K-}}$ est l'ensemble des objets qui n'ont certainement jamais d'attribut dans A et $\overline{(A)^{\prod}_{K+}}$ est l'ensemble des objets qui ont tous leurs attributs à l'exception de A.

On note par $\overline{\mathcal{K}?} == (\mathcal{O}, \mathcal{P}^{-}, (+, -, ?), \overline{R})$ le complémentaire du contexte formel incomplet $\mathcal{K}^{?}$. Il est défini en utilisant la négation de Kleene, à savoir (x, a, +) ∈ $\overline{R}$ si et seulement si (x, a, -) ∈ R, (x, a, -)∈ $\overline{R}$ si et seulement si (x, a, +)∈ R, et (x, a, ?) ∈ $\overline{R}$ si et seulement si (x, a, ?) ∈ R. Il est facile de voir que le lower complétion de $\overline{K?}$ et $\overline{R}$+, le complémentaire de la complétion haute (upper complétion) de $\mathcal{K}^{?}$,et la complétion haute de $\overline{K?}$ et $\overline{R}$-, le complémentaire de la complétion basse (lower complétion) de $\mathcal{K}^{?}$,Nous obtenons deux autre résultats :

Théorème 3 [Ait Yakoub and All, 2017]

⋁ A→⋁B est une implication d'attributs disjonctive certaine si et seulement si $(A)^{\prod}_{K+} \subseteq (B)^{\prod}_{K-}$

Preuve du théorème

D'une part, par le théorème 1, $B^{-} \rightarrow A^{-}$ est une implication d'attributs disjonctive certaine dans $\overline{K?}$ si et seulement si $B^{\Delta}_{K-} \subseteq A^{\Delta}_{K+}$ puisque $\overline{K-}$et $\overline{K+}$ sont respectivement la haute complétion et la basse complétion (the upper ans the lower completions) dans $\overline{K?}$; cela signifie également que ⋁ A→⋁B est une implication d'attributs disjonctives certaine dans$\overline{K?}$.

D'autre part, nous avons : $B\frac{\Delta}{K-} \subseteq A\frac{\Delta}{K+}$ si et seulement si $(A)^{\prod}_{K+} \subseteq (B)^{\prod}_{K-}$.

Théorème 4

$\vee A \rightarrow \vee B$ est une implication d'attributs disjonctive possible si et seulement si $(A)^{\prod}_{K-} \subseteq (B)^{\prod}_{K+}$.

Preuve du théorème

D'une part, par le théorème 2 , $B^{-} \rightarrow A^{-}$ est une implication d'attributs certaine dans $\overline{K?}$ si et seulement si $B\frac{\Delta}{K+} \subseteq A\frac{\Delta}{K-}$. Cela signifie également que $\vee A \rightarrow \vee B$ est implication d'attributs disjonctive possible dans $\overline{K?}$.

D'autre part, $B\frac{\Delta}{K+} \subseteq A\frac{\Delta}{K-}$ est équivalent ($A)^{\prod}_{K-} \subseteq (B)^{\prod}_{K+}$.Par conséquent ,nous obtenons que $\vee A \rightarrow \vee B$ est implication d'attributs disjonctive possible si et seulement si ($A)^{\prod}_{K-} \subseteq (B)^{\prod}_{K+}$

II.5. Conclusion :

Dans ce chapitre, nous avons présenté la théorie de l'analyse de contextes formels incomplets en décrivant d'abord ses fondements mathématiques. Comme nous avons mis l'accent sur les générations des différents contextes formels incomplets possibles, par la suite nous avons présenté le contexte minimal et maximal à l'aide du Lemme1. Enfin nous avons présenté les implications d'attributs conjonctives dans l'analyse de contextes formels incomplets en mettant en évidence les implications d'attributs certaines et possibles et pour cela nous avons utilisé quelques théorèmes qui facilitent la tache de la vérification sans avoir à énumérer tous les contextes formels possibles.

L'analyse de contextes formels incomplets est un domaine très vaste. Ce chapitre n'a été en fait qu'un aperçu sur ce domaine. Par conséquence, la notion que nous avons voulu détailler est celle des implications d'attributs possibles dans un contexte formel incomplet.

Dans le prochain chapitre, nous allons présenter notre contribution qui constitue un pont entre la théorie et la pratique, en présentant un programme informatique qui répond à des requêtes de vérification d’implications possibles et la génération de toutes les implications possibles pour un contexte formel incomplet donné.

Chapitre III

Analyse et conception

III.1. Introduction :

L'analyse de concepts formels à été centré autour de la notion de concepts formels. Ces concepts formels sont extraient à partir d'une relation binaire (contexte formel) entre un ensemble d'objet et un ensemble d'attributs. Comme on là montré dans le chapitre précédent. L'analyse de concepts formels est basée sur des relations Booléennes (ACF classique) peut présenter certaines insuffisances car la réalité présente souvent des situations ou la relation entre l'objet et l'attribut n'est pas renseignée.

Notre contribution consistera à trouver une solution informatique pour une meilleure prise en charge des contextes formels incomplets plus précisément des implications d'attributs conjonctives possibles pour un contexte formel incomplet donné.

Dans le but d'une meilleure organisation et une bonne maîtrise du travail, nous proposons une première contribution qui consiste à vérifier si une implication entre plusieurs attributs saisis par l'utilisateur est possible ou pas, par la suite une seconde contribution consiste à la génération de toutes les implications d'attributs conjonctives possibles et cela pour un contexte formel incomplet donné par l'utilisateur.

III.2. Problématique :

Vérifier si une implication d'attributs est possible dans un contexte formel incomplet exige l'énumération de tous les K contextes formels possibles qui est égal a 2^n avec n le nombre de relations incertaines dans le contexte symbolisées avec le point d'interrogation « ? », par conséquent plus le nombre de relations incertaines est important plus le nombre de contexte généré est grand, ce qui rend la vérification manuelle très difficile voir impossible dans certains cas pour les raisons suivantes :

- ✓ Erreurs de saisi lors du remplacement des relations incertaines « ? »par des « + » ou des « - ».
- ✓ Le cout en termes de temps.
- ✓ Demande beaucoup d'effort et de concentration pour ne pas se tromper lors du remplacement.

III.3. Approche proposé :

Nous avons vu dans le théorème 02 dans le chapitre II comment s'effectue la vérification d'une implication d'attributs conjonctive. Une implication d'attribut est dite possible dans un contexte incomplet $\mathcal{K}^?$ si et seulement si elle est valide dans au moins un contexte formel $\mathcal{K}$ tel que $\mathcal{K}_- \preccurlyeq \mathcal{K} \preccurlyeq \mathcal{K}^+$. Dans ce qui suit nous proposons notre contribution qui consiste en la traduction de ce théorème en solution informatique permettant la vérification et la génération de toutes les implications d'attributs possibles pour chaque contexte formel incomplet.

Rappel du théorème

A→B est une implication d'attributs possible dans $\mathcal{K}^?$ Si et seulement si $A_{K-}^{\Delta} \subseteq B_{K+}^{\Delta}$

III.4. Architecture de l'application :

L'architecture d'un logiciel permet de décrire la manière dont seront agencés les différents éléments d'une application et comment ils interagissent entre eux. Une bonne architecture se définit par sa capacité a prendre en charge des évolutions futures du logiciel en fonction du besoin métier, et aussi par sa simplicité car une architecture trop complexe est souvent source de défaillance et peut créer de la dette technique, de plus une application doit avoir une architecture « compréhensible » pour faciliter sa prise en main pour cela il est nécessaire de respecter les standards afin qu'il soit possible pour une personne connaissant pas le projet d'intervenir.

En fonction des exigences techniques, opérationnelles et fonctionnelles de notre application, nous avons pensé à l'architecture illustrée dans la figure III.1

✓ **Description textuelle de l'architecture de l'application**

Dans un premier temps on a un ensemble de connaissances non structurées qu'on va analyser et transformer en un concept formel qui sera incomplet dans le cas ou il ya manque d'informations (CFI), ensuite à partir de ce contexte formel incomplet conformément aux exigences du théorème 2 du chapitre précédent on génère les deux contextes formels complet optimiste (CFO) et pessimiste(CFP), le premier sera générer en remplaçant tout manque de relation entre objet et attribut « ? » par une présence de relation « + » , le second par absence de relation « - ». L'utilisateur n'aura ensuite qu'à exprimer son besoin sous forme de requêtes soit pour vérifier si une implication est possible pour ce contexte soit pour avoir toutes les implications. Une fois que le résultat est obtenu, il sera interprété par l'utilisateur qui va se charger de prendre la meilleure décision qui lui convient en fonction des résultats obtenus.

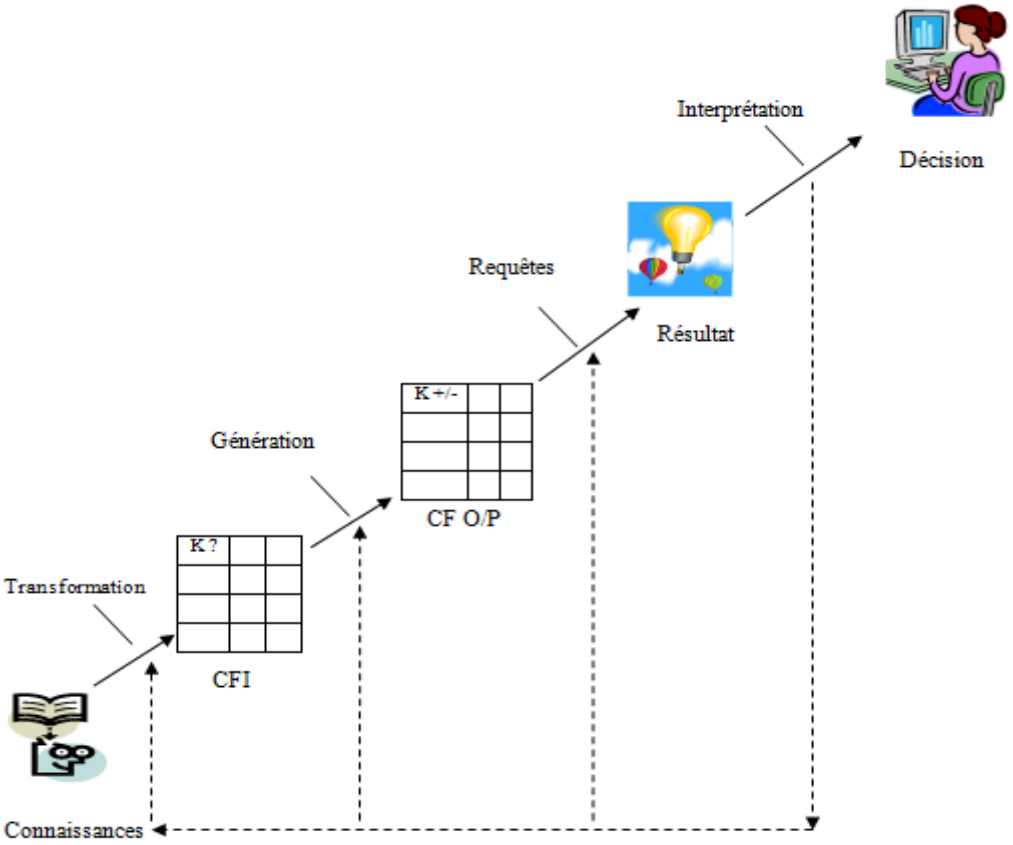


Figure III.1 : Architecture de l'application

III.5. Diagramme de cas d'utilisation :

Le diagramme des cas d'utilisation permet de présenter les acteurs du système, ses grandes fonctions ainsi que les liens existants entre ces fonctions.

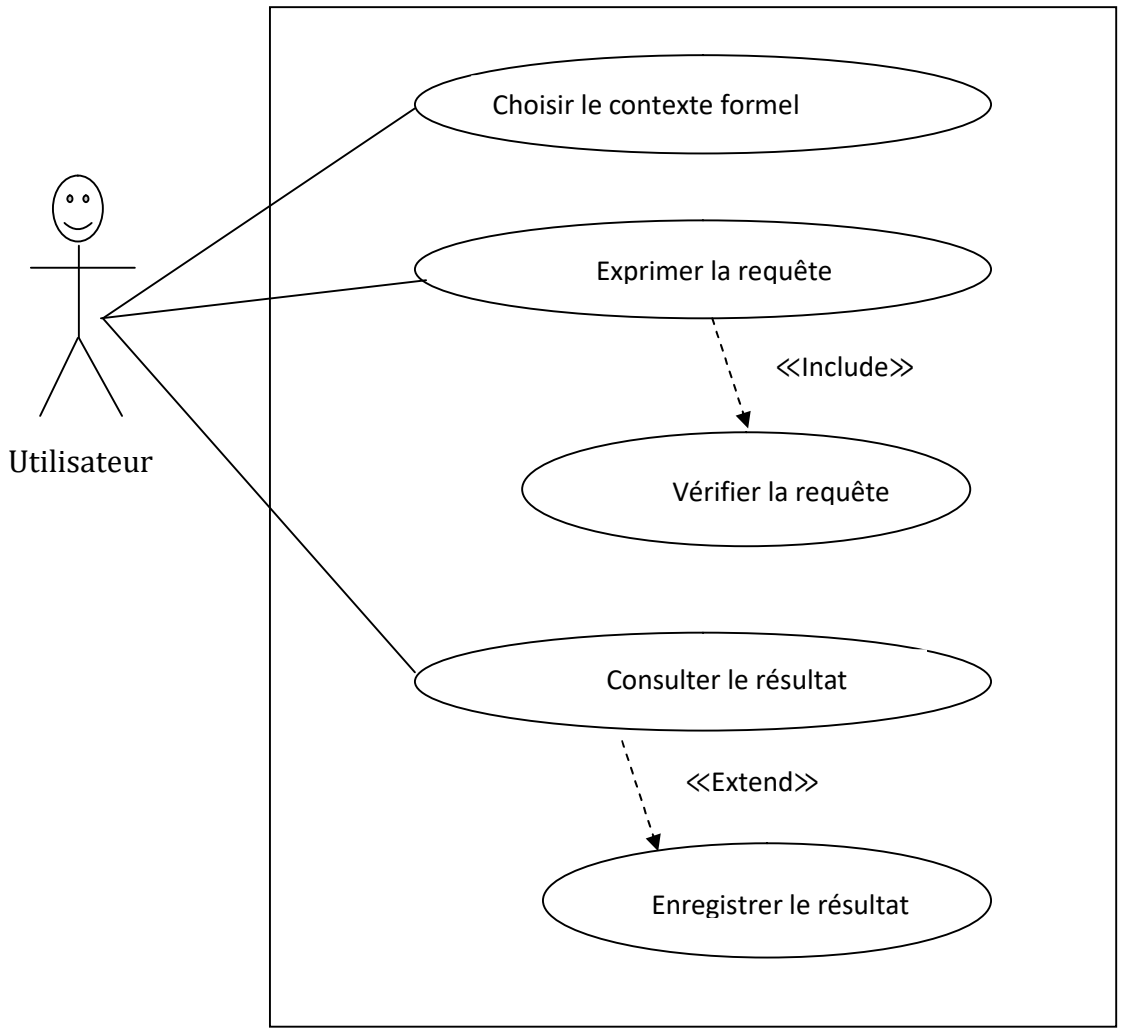


Figure III.2 : Diagramme de cas d'utilisation de l'application

- ✓ **Description textuelle du diagramme de cas d'utilisation**

Dans cette application on a un seul acteur qui est l'utilisateur, il a essentiellement trois actions possibles :

La première action permet de choisir le contexte formel incomplet et le charger en mémoire.

La deuxième action permet à l'utilisateur d'exprimer la requête qui lui convient. Il a la possibilité de tester si une implication d'attribut est possible pour un contexte formel choisi, dans ce cas un teste est effectué pour vérifier si un contexte est bien choisi et un autre teste pour voir si les deux parties prémisse et conclusion sont renseignées, une autre possibilité est celle de générer toutes les implications possibles pour le contexte formel chargé en mémoire.

La troisième action permet d'enregistrer l'ensemble des implications d'attributs possibles générés

III.6. Algorithmes utilisés :

Dans cette section nous allons présenter les deux algorithmes utilisés dans l'application. Le premier est conçu pour vérifier si une implication d'attribut saisie par l'utilisateur est possible ou pas, le second est conçu pour la génération de toutes les implications d'attributs conjonctives possibles d'un contexte formel incomplet donné par l'utilisateur.

Pour faciliter le travail on se base sur le théorème 2 qu'on a démontré dans le chapitre II qui nous permet de restreindre le nombre de contexte formel possible à deux seulement.

III.6.1. Algorithme de vérification d'implications d'attributs conjonctives possibles :

L'algorithme proposé est conçu pour vérifier si une implication d'attribut saisie par l'utilisateur est possible pour un concept formel incomplet choisi aussi par l'utilisateur.

Pour se faire l'utilisateur devra saisir la partie prémisse et conclusion de l'implication.

Le concept formel incomplet est déjà établi et chargé dans la mémoire.

Dans ce qui va suivre on va présenter la classe Valider_implication.

Algorithme valider Implication

Entrée : Le contexte formel pessimiste K^- = (O, P, R)

Le contexte formel optimiste K^+ = (O, P, R)

Sortie : Message « implication possible » / « implication impossible »

```
1.  Debut
2.    Premisse←cg.split(",");
3.    Conclusion←cd.split(",");
4.    Pour i←0 à Premisse.length faire
5.            Pour j←1 à n faire
6.                Si table_moins[0][j]←Premisse[i] faire
7.                        Vect1[l1]←j
8.                            l1++
9.                 Fin Si
10.           Fin Pour
11.   Fin Pour
12.   c←0
13.   Pour z←1 à m faire
14.      Pour c←0 à l1 faire
15.         indice ← Vect1[c]
16.         Si table_moins[z][indice]< > null faire
17.               cpt++;
18.         Fin Si
19.      Fin Pour
20.      Si cpt=l1 Faire
21.                ObjetPremisse[p1]←table_moins[z][0]
22.                    p1++
23.      Fin Si
24.      Pour i←0 à Conclusion.length faire
25.        Pour j←1 à n faire
26.               Si  table_plus[0][j].equals(Conclusion[i]))
27.                           Vect[l]←j
28.                           l++
29.               Fin Si
30.         Fin Pour
31.      Fin Pour
```

```
32.    Pour z←1 à m faire
33.        Pour c←0 à l faire
34.            int indice ←Vect[c];
35.                Si table_plus[z][indice] < > null faire
36.                        cpt++
37.                Fin Si
38.         Fin Pour
39.         Si  cpt=l faire
40.                ObjetConclusion[p]←table_plus[z][0]
41.                 p++
42.         Fin Si
43.     Fin Pour
44.     Pour i←0 à ObjetPremisse.length faire
45.       Si ObjetPremisse[i] < > null faire
46.              cptObj++
47.       Fin Si
48.       Pour j←0 à ObjetConclusion.length faire
49.          Si ObjetConclusion[j] < >null) faire
50.             Si ObjetPremisse[i]=ObjetConclusion[j] faire
51.                    cptEgal++
52.              Fin Si
53.           Fin Si
54.         Fin Pour
55.      Fin Pour
56.      Si cptEgal=cptObj faire
57.        Retourner ("l'implication est possible")
58.      Sinon
59.        Retourner ("l'implication est impossible");}
60.      Fin Si
61. Fin
```

- ✓ **Présentation de l'algorithme**

Dans l'algorithme précédent nous avons présenté la classe ValiderImplication utilisée pour vérifier si une implication saisie par l'utilisateur est possible ou impossible pour un contexte formel incomplet donné. Nous avons utilisé les deux tables table_plus et table_moins pour récupérer les deux contextes formels appelés respectivement pessimiste et optimiste.

Dans le premier on remplace toutes les relations non renseignées '?' par des moins '-', dans le deuxième on les remplace par des plus '+', ces deux tables sont générées dans deux autres classes qu'on présentera par la suite.

1→3 : Récupération des deux cotés prémisses et conclusions saisis par l'utilisateur.

3→32: Après la récupération de la prémisse dans le vecteur prémisse on parcours ce dernier et on compare son contenu avec les attributs de la table table_moins si on a égalité entre les éléments on récupères les indices de ces éléments et on les mets dans le vecteur d'indices vect1, ensuite on parcours les relation entre tous les objets de la table_moins et les indices récupérés dans le vecteur vect1, si on trouve une présence de relation on augmente le compteur cpt, et on récupère les objets qui satisfont la relation et on les mets dans la table Objet_prémisse.

33→44: Après la récupération de la conclusion dans le vecteur conclusion on le parcours et on compare son contenu avec les attributs de la table table_plus si on a égalité entre les éléments on récupères les indices de ces éléments et on les mets dans le vecteur d'indices vect , par suite on parcours les relation entre tous les objets de la table_plus et les indices récupérés dans le vecteur vect si on trouve une présence de relation on augmente le compteur cpt , et on récupère les qui satisfont la relation et on les mets dans la table Objet_conclusion.

45→48 : On compte le nombre d'objets présents dans Objet_conclusion a l'aide de cptObj.

49→56 : On compare les éléments présents dans Objet_premisse et Objet_Conclusion, s'il y a égalité on a incrémente le compteur d'égalité cptEgal.

57→ 61 : On compare cptEgal avec cptObj si on a égalité on affiche le message « Implication possible » dans le cas contraire on affiche « Implication impossible»

Exemple illustratif

Soit la table III.1 suivante qui illustre un contexte formel incomplet et la les tables Table III.2 et Table III.3 qui illustrent respectivement la table_moins et la table_plus dans l'algorithme précédent.

$K^{?}$	a1	a2	a3
x1	?	+	-
x2	+	+	?
x3	-	?	-

Table III.1.contexte Formel incomplet

K^{-}	a1	a2	a3
x1	-	+	-
x2	+	+	-
x3	-	-	-

Table III.2 Table_ plus

K^{+}	a1	a2	a3
x1	+	+	-
x2	+	+	+
x3	-	+	-

Table III.3, Table_ moins

Soit l'implication saisie par l'utilisateur : a1, a2→a3

Prémisse: a1 | a2

Conclusion : a3

Objet_prémisse : x2

Objet_conclusion x2

On a le nombre d'objet dans Objet_prémisse égal au nombre d'objet dans Objet_conclusion si on compare les éléments de ces deux vecteurs on les trouves égaux donc on peut dire que l'implication est possible.

III.6.2. Algorithme de génération de toutes les implications d'attributs conjonctives possibles :

L'algorithme proposé est conçu pour générer toutes les implications d'attributs possibles pour un contexte formel incomplet donné par l'utilisateur.

Le concept formel incomplet est déjà établi et chargé dans la mémoire.

Dans ce qui va suivre on va présenter la classe TouImpPoss.

Algorithme génération implications possibles

Entrée : Le contexte formel pessimiste K^- = (O, P, R)

Le contexte formel optimiste K^+ = (O, P, R)

Sortie : Liste de toutes les implications possibles

```
1.  Debut
2.      Pour j←1 à n faire
3.          attribut←table_plus[0][j1]
4.          Pour l←1 à m faire
5.              Si table_plus[l][j] = null faire
6.                  vect[k] =l
7.                  k1++
8.                  cpt=k1
9.              Fin Si
10.         Fin Pour
11.         Si cpt =0 faire
12.             Pour i←1à n faire
13.                 Si attribut < > (table_moins[0][i1] faire
14.                     Retourner (table_moins[0][i1]+"-->"+attribut)
15.                 Fin Si
16.             Fin Pour
17.         Fin Si
```

```
18.          Pour  i1←1 à n faire
19.            Si attribut<> (table_moins[0][i] faire
20.              Si  table_moins[vect[k]] [i]=null faire
21.                k++
22.                 Si k=cpt faire
23.                     Retourner (table_moins[0][i]+"-->"+attribut)
24.                 Fin Si
25.              Fin Si
26.            Fin Si
27.          Fin Pour
28. Fin
```

- ✓ **Présentation de l'algorithme**

L'algorithme précédent présente une partie de la classe TouImplPoss qui est chargée de générer toutes les implications d'attribut possibles pour un contexte formel incomplet donné.

1→10 : On choisi un attribut à tour de rôle et on enregistre les objets pour lesquels il n'est pas satisfait dans le vecteur Vect et on les comptes a l'aide du compteur Cpt.

11→17 : On vérifie si l'attribut satisfait tout les objets dans ce cas toutes les combinaisons sont possibles avec cette attribut.

18→21 : On vérifie si la combinaison d'attributs en cours de traitement ne satisfait pas les objets enregistrés avec leurs indices dans vect.

22→27 : on compare les deux compteurs K et Cpt, avec K le nombre d'objet non satisfait avec la combinaison d'attributs, s'ils sont égaux l'implication entre la combinaison d'attributs et l'attribut est possible.

- ✓ **Explication de l'algorithme**

Dans cette algorithme nous avons produit toutes les combinaisons possibles avec les attributs présents dans le contexte formel comme partie prémisse et du coté conclusion nous avons pris les attributs un par un et vérifier les combinaisons qui le valide. Pour la vérification nous avons suivi une certaine logique qui permet de réduire le nombre de tests dans le programme afin de réduire le temps d'exécution. Nous avons commencé par prendre attribut par attribut et nous avons vérifié dans la table_plus qui est le contexte optimiste les objets pour lesquels l'attribut n'est pas satisfait par suite on a met ces objets dans un vecteur et on les compter. Il faut noter que si l'attribut a_i en cours de traitement est vérifier pour tout les objets o_i (i=1…n) donc pour toutes les combinaisons existantes l'implication $o_i, o_{i+1},\ldots o_n \rightarrow a_i$ est une implication possible.

Dans la table_moins qui est le contexte pessimiste nous avons pris les combinaisons et vérifier s'ils ne satisfont pas la relation pour les objets précédemment enregistré dans le vecteur, dans ce cas on dit que la combinaison implique l'attribut choisi et on vérifie la combinaison suivante, dans le cas contraire on l'ignore et on passe a la combinaison suivante.

Pour avoir toutes les combinaisons possibles avec les attributs du contexte formel donné, nous nous sommes servi des boucles imbriqués comme illustré dans l'exemple suivant qui produite les toutes les combinaisons possibles pour trois attribut avec cinq le nombre d'attribut dans le contexte :

```
1. Debut
2.         Pour i←1 à 5  faire
3.             Pour j←i+1 à 5 faire
4.                 Pour k←j+1 à 5 faire
5.                     Retourner (i ; j ; k)
6.                 Fin Pour
7.             Fin Pour
8.         Fin Pour
9. Fin
```

Le programme engendre les combinaisons suivantes : 123 ; 124 ; 134 ; 234, les chiffre sont les indices des attributs.

Exemple illustratif

Soit le contexte formel incomplet de la Table III.4 avec les deux contextes pessimiste et optimiste représentés respectivement dans la Table III.5 et Table III.6 :

$K_1^?$	a1	a2	a3
x1	-	+	?
x2	?	+	+
x3	-	?	-

Table III.4 : contexte formel incomplet $k_1^?$

K_1^-	a1	a2	a3
x1	-	+	-
x2	-	+	+
x3	-	-	-

Table III.5 : Table_moins de $K_1^?$

K_1^+	a1	a2	a3
x1	-	+	+
x2	+	+	+
x3	-	+	-

Table III.6 : Table_plus $k_1^?$

- Liste des combinaisons possibles: a1 ; a2 ; a3 ; (a1, a2), (a1, a3), (a2, a3), (a1, a2, a3) ;

- Liste des implications possibles pour ce contexte : a1→a3 ; a1→a2 ; a2→a1 ; a2→a3 ; a3→a1 ; a3→a2 ; (a1;a2)→a3 ; (a1;a3)→a2 ; (a2;a3)→a1.

- Pour calculer le nombre de combinaison possible : 2^n-1*n avec n qui est le nombre d'attribut.

III.6.3. Algorithme production des deux contextes formels optimiste et pessimiste :

Dans les deux algorithmes précédents nous avons fait appel aux deux tables table_plus et table_moins, qui représentent les deux concepts formels complets nécessaires pour appliquer le théorème 2 du chapitre II.

L'algorithme ci-dessus est conçu pour produire les deux contextes formels utilisés dans les traitements précédents. Chaque contexte est conçu dans une classe à part pour faciliter une éventuelle modification. Ces deux contextes sont produits à partir du contexte formel incomplet saisi par l'utilisateur.

Algorithme du contexte formel optimiste

Entrée : Le contexte formel incomplet $K^{?} = (O, P, R)$

Sortie : Le contexte formel optimiste $K^{+} = (O, P, R)$

```
1.  Debut
2.      Pour i←0 à m faire
3.          Pour j←0 à n faire
4.              Si table1[i][j] < > null faire
5.                  Si table1[i][j].equals("?") faire
6.                      table_plus[i][j]←("+")
7.                  Sinon
8.                      table_plus[i][j]=table1[i][j];
9.                  Fin Si
10.             Fin Si
11.         Fin Pour
12.         table_plus[0][0]←"k+"
13.     Fin Pour
14. Fin
```

- ✓ **Explication de l'algorithme**

Cet algorithme est chargé d'appliqué un traitement sur le contexte formel incomplet introduit par l'utilisateur. On parcourt toutes les cases ou les relations entre objet et attribut sont mentionnées, si le contenu de la case qui est un caractère est égale au « ? » on le changer par le « + », sinon on garde le contenu tel qu'il est.

On a affecté la chaine « K+ » à la case de la ligne 0 et de la colonne 0 pour dire que c'est le contexte optimiste.

Exemple

Soit le contexte formel incomplet de la table III.7 introduit par l'utilisateur avant d'appliquer l'algorithme de la classe K_Plus, et la table III.8 qui représente le résultat :

k ?	a1	a2	a3
x1	+	?	-
x2	-	+	?
x3	?	-	+

Table III.7: contexte formel incomplet

k+	a1	a2	a3
x1	+	+	-
x2	-	+	+
x3	+	-	+

Table III.8 : contexte formel optimiste

Algorithme du contexte formel pessimiste

Entrée : Le contexte formel incomplet $K^{?}$ = (O, P, R)

Sortie : Le contexte formel pessimiste K^{-} = (O, P, R)

```
1.  Debut
2.      Pour i←0 à m faire
3.          Pour j←0 à n faire
4.              Si table1[i][j] < > null faire
5.                  Si table1[i][j].equals("?") faire
6.                      table_plus[i][j]←("-")
7.                  Sinon
8.                      table_plus[i][j]=table1[i][j];
9.                  Fin Si
10.             Fin Si
11.         Fin Pour
12.         table_plus[0][0]←"k-"
13.     Fin Pour
14. Fin
```

- ✓ **Explication de l'algorithme**

Cette algorithme est chargé d'appliqué un traitement sur le contexte formel incomplet introduit par l'utilisateur. On parcourt toutes les cases ou les relations entre objet et attribut sont mentionnées, si le contenu de la case qui est un caractère est égale au « ? » on le changer par le « - », sinon on garde le contenu tel qu'il est.

On a affecté la chaine « K- » à la case de la ligne 0 et de la colonne 0 pour dire que c'est le contexte pessimiste.

Exemple

Soit le contexte formel incomplet de la table III.7 introduit par l'utilisateur avant d'appliquer l'algorithme de la classe K_moins, et la table III.10 qui représente le résultat **:**

k-	a1	a2	a3
x1	+	-	-
x2	-	+	-
x3	-	-	+

Table III**.10** : contexte formel pessimiste

III.7. Conclusion :

Dans ce chapitre nous avons proposé une approche de vérification et génération des implications d'attributs conjonctives possibles dans un contexte formel incomplet, et cela pour faire face à l'incomplétude et au manque d'informations. Cette approche consiste essentiellement à l'application d'un théorème mathématique sur le contexte formel incomplet, comme on la montré dans l'architecture de l'application, ce dernier servira à produire deux autre contextes formels complets, le premier appelé optimiste le second pessimiste en appliquant des algorithmes définis et qui servirons de base d'informations pour la vérification et la génération des implications possibles.

Cette approche représente un pont entre les mathématiques et l'informatique permettant ainsi un gain de temps et d'effort et plus de précision dans les résultats qui seront une source de décisions.

Dans le chapitre suivant nous allons présenter les outils informatiques utilisés ainsi que le langage de programmations qui a servi à la codification. Par la suite nous montrerons un exemple d'application illustré avec des captures d'écran du programme.

Chapitre IV

Réalisation

IV.1. Introduction :

Après avoir présenté dans les chapitres précédents les notions nécessaires à l'accomplissement de notre travail, ce chapitre est maintenant l'occasion de présenter notre contribution à savoir «La vérification et la génération de toutes les implications d'attributs conjonctives possibles pour un contexte formel incomplet donné».

Dans le présent chapitre, nous commencent par montrer l'environnement technique de développement, ensuite les techniques d'implémentations, Nous présenterons aussi l'interface développée accompagnée d'un exemple pour illustrer le fonctionnement de notre application.

IV.2. Environnement technique de développement :

Notre prototype de système a été réalisé sur un PC ayant les caractéristiques suivantes :

- Un microprocesseur Pentium 4, fréquence de l'horloge 2.00 GHz.
- mémoire RAM de 4 Go.
- disque dur 500 Go.
- Ecran LCD couleur 15 pouces.
- Avec Windows 10 professionnel comme système d'exploitation, eclipse comme environnement de développement, et Java comme langage de programmation.

a. Présentation du Langage de programmation utilisé (java)

Java est un langage de programmation informatique orienté objet crée par James Gosling et Patrick Naughton, employees de Sun Microsystems, avec le soutien de Bill Joy (cofondateurde SunMicrosystems en 1982), présenté officiellement le 23 mai 1995 au SunWorldJava possède un certain nombre de caractéristiques qui ont largement contribué à son énorme succès :

- Java est interprété : La source est compilé en pseudo code ou byte code puis exécuté par un interpréteur Java « la Java Virtual Machine (JVM) ». Ce concept est à la base du slogan de Sun pour Java : WORA (Write Once, Run Anywhere : écrire une fois, exécuter partout). En effet, le byte code, s'il ne contient pas de code spécifique à une

  plate-forme particulière peut être exécuté et obtenir quasiment les même résultats sur toutes les machines disposant d'une JVM.

- Java est portable (Il est indépendant de toute plate-forme) : il n'y a pas de compilation spécifique pour chaque plate forme. Le code reste indépendant de lamachine sur laquelle il s'exécute. Il est possible d'exécuter des programmes Java sur tous les environnements qui possèdent une Java Virtual Machine. Cette indépendance est assurée au niveau du code source grâce à Unicode et au niveau du byte code.

- Java est orienté objet : Comme la plupart des langages récents, Java est orienté objet. Chaque fichier source contient la définition d'une ou plusieurs classes qui sont utilisées les unes avec les autres pour former une application. Java n'est pas complètement objet car il définit des types primitifs (entier, caractère, flottant, booléen,...).

- Java est simple : Le choix de ses auteurs a été d'abandonner des éléments mal compris ou mal exploités des autres langages tels que la notion de pointeurs (pour éviter les incidents en manipulant directement la mémoire), l'héritage multiple et la surcharge des opérateurs, ...

- Java est fortement typé : Toutes les variables sont typées et il n'existe pas de conversion automatique qui risquerait une perte de données. Si une telle conversion doit être réalisée, le développeur doit obligatoirement utiliser un cast ou une méthode statique fournie en standard pour la réaliser.

- Java assure la gestion de la mémoire : L'allocation de la mémoire pour un objet est automatique à sa création et Java récupère automatiquement la mémoire inutilisée grâce au « garbage collector » qui restitue les zones de mémoire laissées libres suite à la destruction des objets.

b. Présentation de l'environnement de développement utilisé (eclipse)

Eclipse est un environnement de développement libre. Le projet a été initié par IBM pour remplacer, en utilisant Java, l'IDE Visual Age, basé sur Smalltalk. Dès l'origine du projet, IBM a voulu offrir une solution multiplateforme, pouvant être exécutée sur les différents systèmes d'exploitation de ses clients. De même le projet s'est voulu extensible par le biais de plugins.

Le nom serait un jeu de mots : le créateur de Java est Sun (en français : « soleil »), concurrent qu'IBM semble vouloir « éclipser ».

En juin 2007, parait la version 3.3 appelée Europa. 310 développeurs répartis dans 19 pays ont écrit les 17 millions de lignes de codes qui la composent. Cette version porte officiellement 21 projets.

La version 3.5 appelée Galileo, parue en juin 2009 porte en outre cette fois-ci 33 projets internes allant de modélisation à l'analyse des performances. Plus de 380 membres appartenant à 44 organisations différentes ont contribué à l'élaboration de cette version comportant 24 millions de lignes de code. Six versions de test et d'évaluation, appelées Milestone étaient déjà parues, la première en août 2008, la seconde en septembre 2008, puis Novembre 2008 et Décembre 2008 et enfin la dernière en février 2009 puis Mars 2009.

La version 3.6, appelée Helios, sortie en juin 2010, est déclinée en 12 paquetages, selon les usages pour le développement (C/C++, JavaScript, PHP,...). Elle s'appuie sur 77 projets. Les principales évolutions relevées comprennent : un nouveau paquetage adapté au

développement en C/C++) dans l'environnement système d'exploitation Linux, une nouvelle gestion des plugins, le support de Git, le support des dernières évolutions Java (dont Servlet 3.0, JPA 2.0, JSF 2.0,EJB 3.1), une amélioration du support JavaScript (en mettant en place un cadre pour intégrer des débogueurs tels que Rhino ou Firebug), Eclipse Xtext 1.0, (environnement pour créer des langages spécifiques -domain specific languages, DSL-), une nouvelle version de Acceleo 3.0 (OMG Model-to-text -MTL-). La version actuelle, appelée Photon, est sortie le 14 Décembre 2018.

IV.3. Techniques d'implémentation :

- **Interface graphique**

Pour le développement de notre interface graphique nous avons utilisé l'API SWING.Swing propose de nombreux composants dont certains possèdent des fonctions étendues, une utilisation des mécanismes de gestion d'événements performants et une apparence modifiable à la volée (une interface graphique qui emploie le style du système d'exploitation Windows ou Motif ou un nouveau style spécifique à Java nommé Metal).

Les composants Swing forment une nouvelle hiérarchie parallèle à celle de l'AWT. L'ancêtre de cette hiérarchie est le composant JComponent. Presque tous ces composants sont écrits en pur Java : ils ne possèdent aucune partie native sauf ceux qui assurent l'interface avec le système d'exploitation : JApplet, JDialog, JFrame, et JWindow. Cela permet aux composants de toujours avoir la même apparence quelque soit le système sur lequel l'application s'exécute.

SWING possède plusieurs package, parmi ces package nous avons utilisé :

- Javax.swing.JFram : Classe pour la création de fenêtre
- Javax.swing.JPanel : Classe pour création de panel

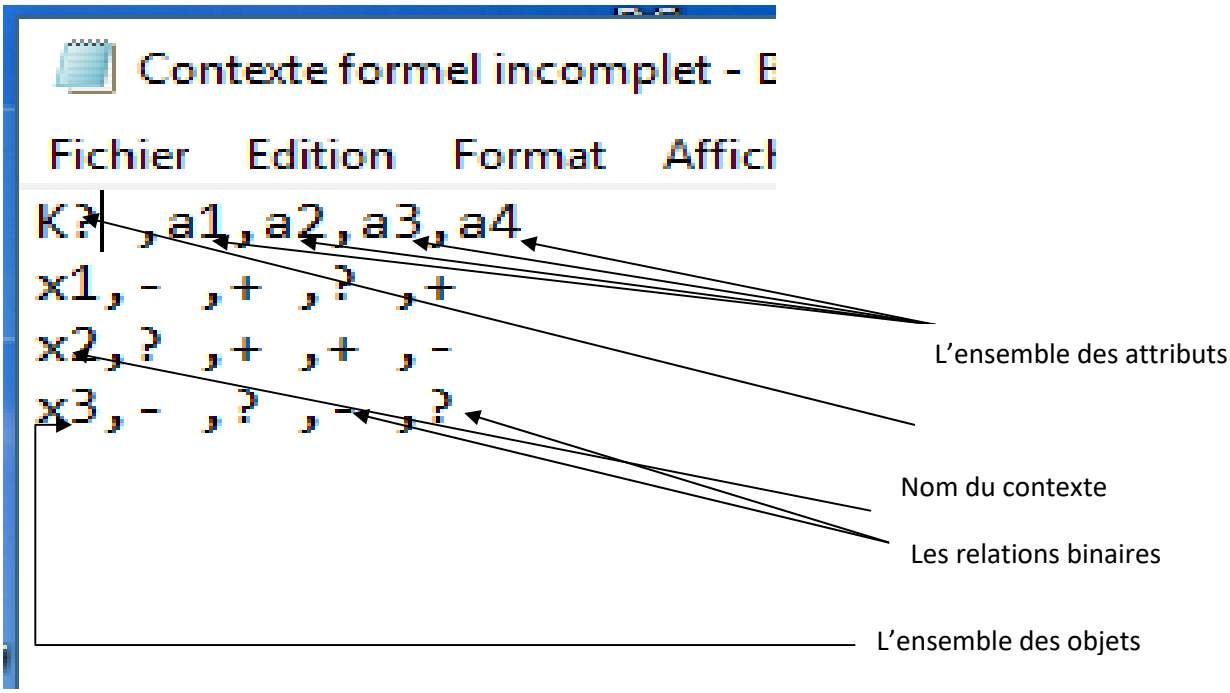
- java.io.BufferedReader : Classe pour la lecture a partir d'un fichier externe
- java.io.BufferedReader : Classe pour l'écriture dans un fichier externe
- javax.swing.event : Classes et interfaces pour les événements spécifiques à Swing.
- javax.swing.text Classes et interfaces de bases pour les composants manipulant du texte.
- javax.swing.JFileChooser : Classe pour la sélection du fichier
- java.io.IOException : Classe pour la gestion des exceptions
- javax.swing.JButton : Classe pour la création des boutons

IV.4. Format d'introduction du contexte formel incomplet :

La structure du fichier que nous avons utilisé pour exprimer un contexte formel incomplet est comme suite :

- La case une de la ligne une contient le nom du contexte formel incomplet symbolisé par « $K^?$ ».
- A partir de la deuxième case de la ligne une on trouve l'ensemble des attributs séparés par des virgules.
- A partir de la deuxième case de la colonne une on trouve l'ensemble des objets.
- L'intersection entre ligne et objet est la relation symbolisé par un « + » dans le cas de sa présence et par un « -» dans le cas contraire, si la relation n'est pas renseignée on met « ? ». L'ensemble des relations est séparé par des virgules.

La figure IV.1 représente une capture d'écran d'un contexte formel incomplet que nous avons sauvegardé dans un fichier .TXT



Colonnes

Lignes

K ?	a1	a2	a3	a4
x1	-	+	?	+
x2	?	+	+	-
x3	-	?	-	?

Figure IV.1 : Représentation d'un contexte formel incomplet dans un fichier

IV.5. Description des interfaces de notre logiciel :

c. Interface principale

Cette fenêtre est constituée de deux parties

- Première partie : celle qui permet à l'utilisateur choisi le fichier TXT qui contient le contexte formel incomplet qu'il souhaite traiter.
- Deuxième partie : celle qui permet à l'utilisateur d'exprimer son choix soit de vérifier la possibilité d'une implication d'attribut, comme il peut choisir de générer toutes ces dernières

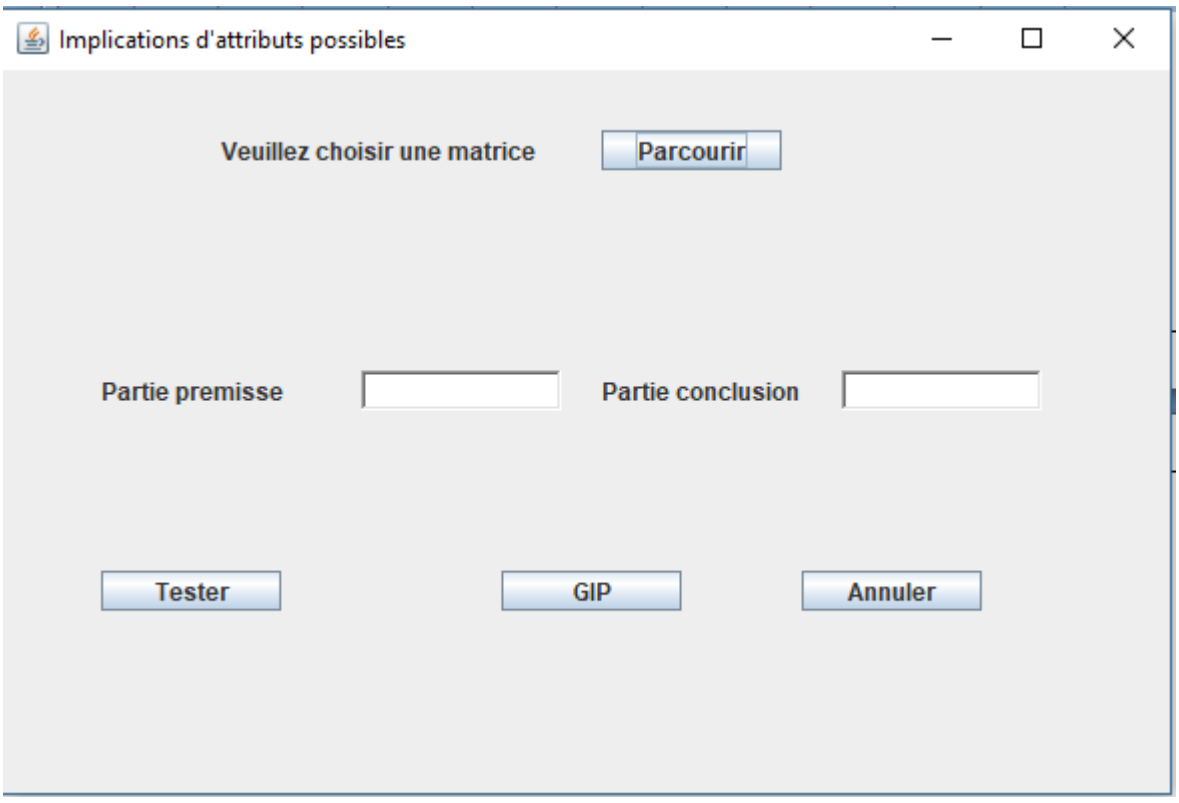


Figure IV.2 : La fenêtrer principale du logiciel

- Parcourir : ce bouton sert a choisir le fichier TXT a traiter.

- Objet prémisse: cette entrée set a saisir les attributs de la partie prémisse sous la forme (a1, a2).
- Objet conclusion : cette entrée sert a saisir l'attribut de la partie conclusion
- Tester : ce bouton sert à vérifier si l'implication d'attribut saisie dans les deux champs précédent est possible.
- GIP : ce bouton sert à générer toutes les implications d'attributs possibles pour le contexte formel choisi précédemment.
- Annuler : ce bouton sert à affecter la valeur nulle aux deux champs de l'implication et au Path du fichier a sélectionner.

Dans ce qui va suivre nous allons dérouler l'application en utilisant un contexte formel incomplet nommé « fichier » enregistré sur notre ordinateur pour voir le fonctionnement de l'application. En appuyant sur le bouton Parcourir on aura la fenêtre décrite dans la figure IV.3. qui permet de choisir un contexte formel par conséquent on récupère le Path du fichier.

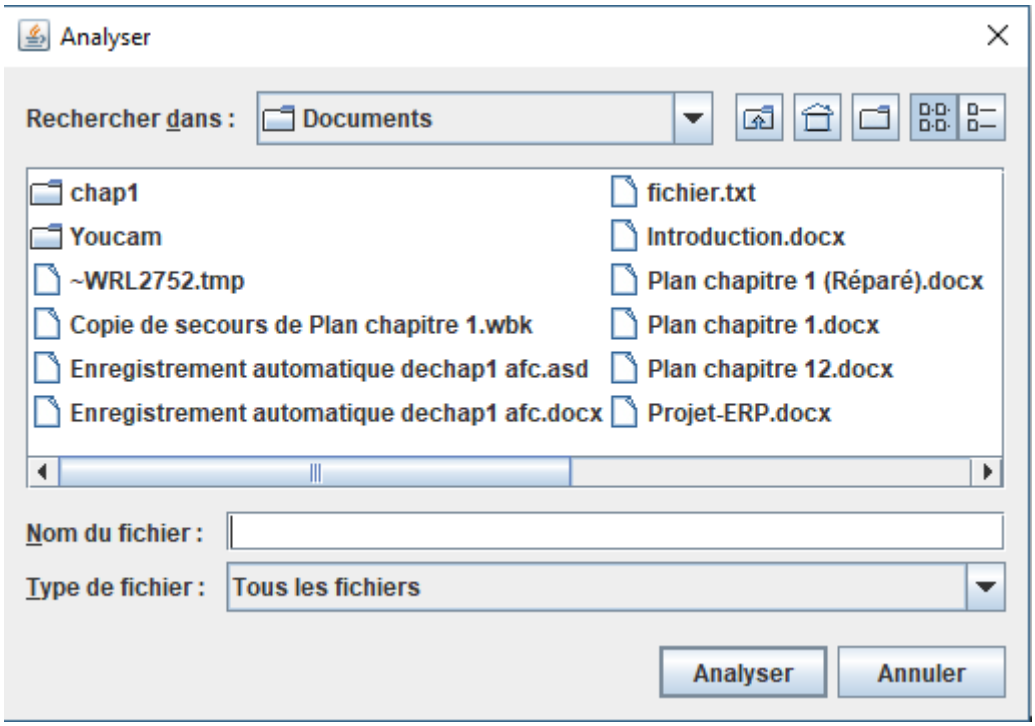


Figure IV.3 : Fenêtre pour la récupération du fichier qui contient le contexte formel incomplet

Une fois le fichier choisi on revient a la fenêtre principale pour tester si une ou plusieurs implications d'attributs conjonctives est possibles, ou générer toutes les implications d'attributs possibles

- Premier cas tester une implication d'attribut conjonctive :

Pour tester si une implication d'attribut est possible on rempli les deux champs prémisse et conclusion et clique sur le bouton « Tester » comme on le montre dans la figure suivante :

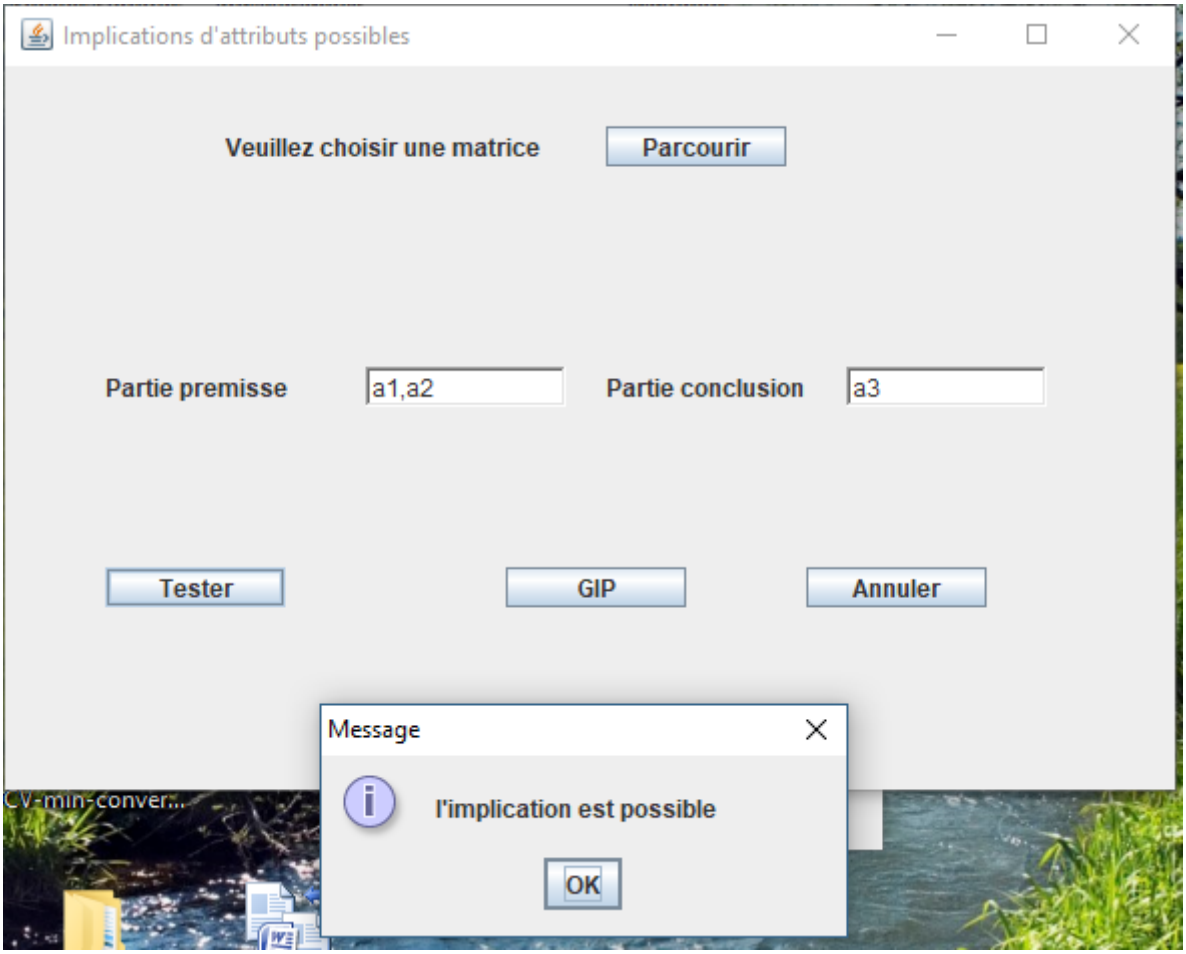


Figure IV.4 : Fenêtre montrant le résultat d'un teste

- Deuxième cas générer toutes les implications d'attributs conjonctives possibles :

Pour se faire on clique sur le bouton « GIP » après bien sûr le choix du contexte formel comme illustre la figure IV.5.

Figure IV.5 : Fenêtre montrant l'ensemble des implications d'attributs possibles

On a la possibilité de sauvegarder le résultat ou de fermer la fenêtre. Si on choisi de sauvegarder, une autre fenêtre va s'ouvrir pour nous demander l'emplacement de sauvegarde, une fois sauvegardé on peut accéder directement la liste des implications d'attributs possibles enregistré dans un autre fichier TXT.

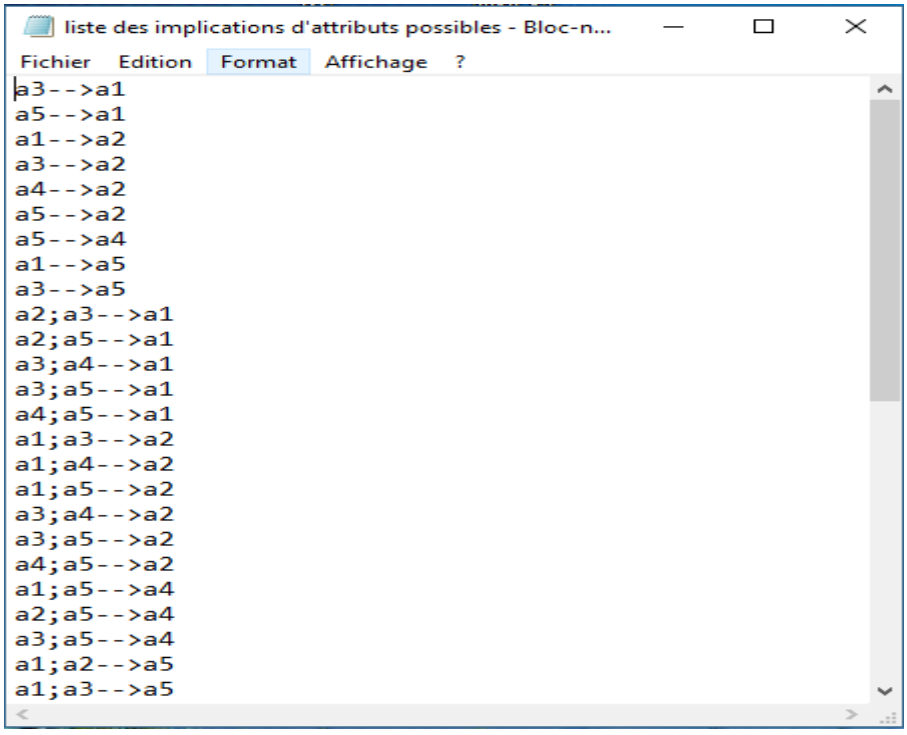


Figure IV.6 : Fichier TXT contenant l'ensemble de toutes les implications d'attributs possibles

IV.6. Conclusion :

Cette partie à été l'occasion de présenter notre contribution ainsi que l'aspect pratique de notre système. Nous avons montré les outils nécessaires pour la réalisation de ce travail ainsi que le langage de programmation et l'environnement du développement.

Nous avons montré aussi le fonctionnement principal de l'application qui est la prise en charge d'un contexte formel incomplet introduit par l'utilisateur via un fichier .TXT pour valider et générer des implications d'attributs possibles toute en donnant un exemple illustratif, accompagné par des captures d'écrans pour chaque phase du traitement

Conclusion et perspectives

Conclusion et perspectives

Les travaux effectués dans le cadre de ce mémoire porte sur l'analyse concepts formels, plus précisément il s'agit de la vérification et la génération des implications d'attributs possibles dans un contexte formel incomplet. La vérification d'une implication d'attributs possible et la génération de toutes les implications d'attributs possibles se font a partir des données choisis par l'utilisateur.

Il existe dans la littérature plusieurs approches définissant l'incomplétude. Certaines de ces approches sont basées sur la théorie des ensembles incomplets [Obiedkov 2002] où la notion de complétude est maintenant représentée par un état de renseignement, c'est-à-dire l'information est complètement renseigner ou pas.

Le premier apport consiste à appliquer le théorème mathématique sur les contextes formels incomplets afin d'avoir une structure complète.

Le deuxième apport consiste à la traduction de l'apport mathématique en algorithmes de génération des contextes formels optimiste et pessimiste, et aux traitements effectués sur ces derniers afin de générer et de valider les implications d'attributs conjonctives possibles. Ces traitements sont effectués par l'utilisateur à l'aide des interfaces que nous avons aussi programmées.

Le sujet abordé dans ce mémoire ouvre diverses perspectives, nous en présentons quelques une.

- ✓ La validation et la génération de toutes les implications d'attribut disjonctives pour des contextes formels incomplets.

- ✓ La complexité algorithmique de notre application est exponentielle donc elle peut toujours être améliorée et rendre les algorithmes plus efficaces surtout en ce qui concerne le nombre d'attribut prit en charge.
- ✓ Compresser la base des implications conjonctives possibles générées à une base minimale en éliminant les implications qu'on peut déduire.
- ✓ Etudier la possibilité de produire des treillis de Galois pour des contextes formels incomplets.

Bibliographie

[Agrawal, 1993]: R. Agrawal, T. Imielinsky, and A. Swami: "Mining association rules between sets of items in large databases". In proceeding of the 93 ACM SIGMOD international Conference on management of Data (SIGMOD'93), pp 207-216, 1993.

[Agrawal et Srikant, 1994] : R. Agrawal and R. Srikant. Fast algorithms for mining Association rules in databases. in Proceeding of the 20th international conference on very large data Bases (VLDB'94), pages 478 _ 499. Morgan Kaufmann, septembre 1994.

[Djouadi, 2009]: Interval-valued fuzzy formal concept analysis. In ISMIS 09 : Proc. Of the 18th International Symposium on Foundations of Intelligent Systems,pages 592-601, Berlin ,Heidelberg, 2009. Springer-Verlag.

[Dubois, 2000] : D. Dubois, H. Prade. Fundamental of fuzzy sets. In: Kluwer Academic Publisher, 2000.

[Ganascia, 1993]: TDIS : an Algebraic Formalization », Proceedings of IJCAI'93, vol. 2, 1993, p. 1008-1013

[Ganter, 1999]:B. Ganter, R. Wille: "Formal Concept Analysis". Mathematic foundation. Springer- verlag, Berlin, 1999.

[Ganter and Wille, 1999] : Ganter, B. and Wille, R. (1999). Formal Concept Analysis.Springer, mathematical foundations edition.

[Obiedkov, 2002]: Modal logic for evaluating formulas in incomplete contexts. In G.Angelova U.Priss, D. Corbett, editor ,conceptual structures: integration and Interfaces, Volume 2393 of LINCS, page, 314-325. Springer, 2002

[Pasquier, 2000] : N. Pasquier: "Algorithmes d'extraction et de réduction des règles d'association dans les bases de données". Université Clermont- Ferrand II, Ecole Doctorale, sciences pour l'ingénieur de Clermont- Ferrand, 2000.

[Priss, 2005] : Priss, U. (2005). *Linguistic applications of formal concept analysis*. In [Ganter et al., 2005a], pages 149–160.

[Burmeister and Holzer, 2005]: Treating incomplete knowledge in formal concept analysis.In B.Ganter, G.Stumme , and R.wille ,editors,Formal concet analysis page 114-126.Sprnger,2005

[Seddoud, 2011] : F.Seddoud : "Construction du treillis de concepts formels pour des contextes à intervalle de vérité à partir d'implications résiduées".Mémoire de magister, 2011.

[Wille, 1982]: R. Wille : "Restructuring lattice theory: An approach based on hierarchies of concepts". In Rival, Dordrecht, Boston, pp 320-350,1982.

[Wille, 1996] : Wille, R. (1996). Restructuring mathematical logic : an approach based on peirce's 281, New pragmatism. In Ursini, A. and Agliano, P., editors, Logic and algebra, pages 267– York. Marcel Dekker