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 certaines.

Proposé et dirigé par :

M me Ait-Yakoub Zina.

Réalisé par :

M elle Djebra Kenza.

M elle Rabahi Lynda.

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.

Remerciement

Nous remercions le bon Dieu de nous avoir mis sur la voie du savoir.

Nous remercions vivement notre promotrice Mme Ait_Yakoub Zina qui nous a encadrées pendant la réalisation de ce mémoire, ainsi pour sa constante disponibilité, pour ses conseils et son soutien tout au long de notre travail.

Nos vifs remerciements vont également aux membres de jury qui nous font l'honneur de juger notre travail.

Nos plus vifs remerciements vont aussi à tous les enseignants du département informatique de la faculté de Génie Electrique et Informatique de l'université MOULOUD MAMMERI pour leurs disponibilités, leurs conseils tout au long de notre cursus au sein de ce département.

Dédicaces

Je dédie ce modeste travail :

A tous ceux qui ont contribué de prés ou de loin pour réaliser ce travail,

A mes très chers parents en particulier qui ont toujours cru en moi et qui m'ont beaucoup aidée à surmonter mes inquiétudes,

A mes chers frères Lyes et Toufik ainsi qu'à toute ma famille,

A mes cousines Samia, Hanane, Chahinez,

A ma chère binôme qui a pu me supporter tout au long de ce travail, c'était un grand plaisir d'avoir partagé toutes ces années avec,

A tous mes amis Yasmine, Sylia, Saliha, Lamia, Kamilia , Dyhia, Rabie, Jugurtha, Azzedine,

A mes petites adorées Wissam, Wiam et Yasmine.

Kenza.

Dédicaces

Je dédie ce modeste travail :

A tous ceux qui ont contribué de prés ou de loin pour réaliser ce travail,

A la mémoire de mon grand père Tahar,

A ma chère grand-mère, que je respecte tant,

A mes très chers parents auxquels je souhaite une longue vie pleine de bonheur,

A mes chères sœurs, Mouna, Lydia, Lilia, Leiticia surtout mon petit ange Lila ainsi qu'à toute ma famille,

A ma chère binôme pour son encouragement et sa patience pendant les moments difficiles,

A tous mes amis, sans citer leurs noms pour ne pas oublier personne,

A tous les étudiants du département Informatique.

Lynda.

Sommaire :

Chapitre III : Analyse & Conception

Chapitre IV : Réalisation & application

Liste des tableaux:

Liste des figures :

Introduction générale:

L'Analyse de Concepts Formels (ACF) (appelée aussi Analyse Formelle de Concepts (AFC)) est un formalisme mathématique pour l'analyse de données, la représentation de connaissances et la visualisation de connaissances. L'idée de base de l'ACF est d'extraire des concepts regroupant des objets et leurs propriétés/attributs à partir de données et de construire une hiérarchie à partir de ces concepts, appelée treillis de concepts. [19]

Classiquement, l'ACF s'applique à une relation binaire représentée par un contexte formel. Un objet satisfait un attribut ou ne le satisfait pas. Au cours des dernières années, l'ACF a été appliquée dans de nombreuses disciplines comme la psychologie, la sociologie, l'anthropologie, la médecine, la biologie, la linguistique, etc. Dans de tels cas, l'ACF traite inévitablement des structures d'information relationnelle (contextes formels) issues de l'investigation humaine (jugement, observation, mesure,… etc.). De pareilles informations sont inévitablement entachées d'incertitude et/ou d'imprécision et/ou d'incomplétude. Des recherches récentes ont été initiées afin de prendre en considération des contextes formels contenant de pareilles informations. [04] [21] [22]

Dans le cadre de ce mémoire, nous allons traiter le cas de l'information incomplète qui correspond à une information partiellement informée, où l'ignorance totale, le cas ou on ne sait pas si l'objet possède l'attribut ou non, ce cas à été considéré par Obiedkov [07] en utilisant la logique modale et par Burmeister et Holzer [08] en utilisant la logique à trois valeurs (Kleene 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, nous allons nous intéresser aux implications résultantes par ce type de contexte, à savoir les implications d'attributs possibles et certaines [20] [23] [24]. Cependant, notre travail consiste en premier lieu à implémenter une méthode qui permet de valider les implications d'attributs introduites par l'utilisateur, en second lieu à mettre en œuvre une approche qui permet de générer toutes les implications d'attributs conjonctives certaines à partir d'un contexte formel incomplet.

Pour mener à terme notre travail, nous le répartissons de la manière suivante :

Après avoir introduit notre thématique, le premier chapitre expose les notions de bases permettant de bien définir l'analyse de concepts formels (ACF). En premier temps l'ACF classique permet de représenter les données d'un contexte formel (k) sous forme binaire tel que : un attribut satisfait un objet ou ne le satisfait pas.

Ce cadre d'analyse semble restrictif pour l'étude des données complexes ce qui a engendré un contexte formel incomplet qu'on va exposer dans le deuxième chapitre où apparaît notre problématique qui repose sur deux points essentiels :

-Vérifier la certitude d'une implication d'attributs conjonctives obtenue à partir d'un contexte formel incomplet.

-Trouver une façon pour générer toutes les implications conjonctives certaines à partir d'un contexte formel incomplet.

Le troisième chapitre représente notre contribution qui consiste à proposer un algorithme pouvant vérifier la certitude des implications d'attributs conjonctives obtenues à partir d'un contexte formel incomplet. Puis proposer un autre algorithme qui permet de générer toutes les implications d'attributs certaines.

Le quatrième chapitre comporte quand à lui la présentation de l'environnement dont lequel notre application a été réalisé, les outils utilisés et quelques interfaces de notre application.

On termine ce mémoire avec une conclusion générale et évoquer quelques perspectives.

Chapitre I : Analyse de concepts formels

Sommaire :

I-1)-Introduction :

L'analyse de concepts formels (ACF) est une méthode d'analyse de données centrée sur l'humain, définie comme un moyen de d'exploration et de discussion. L'ACF a été lancé comme une tentative à restructurer la théorie mathématique d'une manière qui facilite à la fois la communication et l'exploitation sur la théorie mathématique à un public non mathématique plus large.

L'ACF est spécialisée dans l'extraction d'un ensemble de données visant à identifier des regroupements objets/attributs, appelés concepts formels, et à ordonner ces regroupements sous la forme de treillis de concepts qui est une vue particulière sur les données, qui révèle des règles d'implications entre groupes de liens.

Nous présentons dans ce chapitre une vue générale de l'analyse de concepts formels en commençant par son origine, une présentation de cette dernière, quelques rappels mathématiques à revoir ensuite les concepts de base de l'ACF. Ce cadre d'analyse est appliqué à différentes tâches incluant plusieurs domaine qu'on va citer à la fin de ce chapitre.

I-2)- Origine et contexte philosophique :

L'Analyse de Concepts Formels (ACF) [14] a été présentée comme un domaine de mathématiques appliquées qui consiste à restructurer la théorie des treillis [12] afin de faciliter son utilisation dans des applications du monde réel et de permettre l'interprétation de ses notions en dehors du cadre théorique aussi bien par des mathématiciens que par des non-mathématiciens. L'objectif à travers la mise en place de l'ACF est "d'atteindre une théorie structurée qui expose les pensées formelles selon des interprétations significatives et permettre ainsi des communications et des discussions critiques de leurs contenus" [07]. Pour cela l'ACF a été centrée autour de la notion de concept qui, du point de vue philosophique, est considéré comme l'unité de base de la pensée humaine. De manière informelle, un concept peut être défini comme un groupement d'individus et de leurs propriétés communes. [07]

I-3)- Présentation intuitive de l'analyse de concepts formels :

L'Analyse de Concepts Formels (ACF) est une méthode d'analyse de données spécialisée dans l'extraction d'un ensemble ordonné de concepts au sein d'un ensemble de données [01]. Cet ensemble de données, appelé un contexte formel, est composé d'objets décrits par des attributs, qui peuvent être représentés sous la forme d'un tableau. La case (i,j) est marquée d'une croix (x) si l'objet i possède l'attribut j, sinon la case est vide. Dans l'exemple suivant, on reprend le contexte formel donné dans [02].

Exemple:

Contexte formel

R	Homme	Femme	Mère	Père	Parent
John	×				
Maria		×			
Peter	×			×	×
Clara		×	×		×

Table 1.1 : Contexte formel.

I-4)- Rappels mathématiques :

Définition 1 (Relation binaire) :

Une relation binaire R entre deux ensembles O et P est un ensemble de couples d'éléments (x; a) tels que x $\in$ O et a $\in$ P, autrement dit un sous ensemble du produit Cartésien O×P. (x; a) $\in$ R (aussi noté par xRa) signifie que l'élément x est en relation R avec l'élément a.

Définition 2 (Relation d'ordre) :

Une relation binaire R sur un ensemble E est dite relation d'ordre partiel (ou simplement relation d'ordre) sur E si elle vérifie les conditions suivantes pour tout x, y et z éléments de E :

_ (x; x) $\in$R (R est réflexive)

_ Si (x; y) $\in$ R et (y; x) $\in$R alors x = y $\rightarrow$ (R est antisymétrique)

_ Si (x; y) $\in$ R et (y; z) $\in$ R alors (x; z) $\in$ R $\rightarrow$(R est transitive)

Une relation d'ordre R est souvent notée par $\leq$.

Définition 3 (Ensemble ordonné) :

Un ensemble partiellement ordonné (ou simplement ensemble ordonné) est un couple (E; $\leq$) où E est un ensemble et '$\leq$' est une relation d'ordre sur E.

Dans un ensemble ordonné (E; $\leq$), deux éléments x et y de E sont dits comparables lorsque x $\leq$ y ou y $\leq$ x, autrement ils sont dits incomparables. Un sous ensemble de (E; $\leq$) dans lequel tous les éléments sont comparables est appelé chaîne. Un sous ensemble de (E; $\leq$) dans lequel tous les éléments sont incomparables est appelé anti-chaîne.

I-5)- Théorie de l'analyse de concepts formels :

I-5-1) Contexte formel :

Un contexte formel est un triplet K = (O;P;R) où O est un ensemble d'objets, P est un ensemble d'attributs (propriétés) et R est une relation binaire entre O et P appelée relation d'incidence de K et vérifiant (R $\subseteq$ O×P). Un couple (x; a) $\in$ R (noté aussi xRa) signifie que l'objet{x $\in$O} possède (satisfait) l'attribut {a $\in$ P}.

Un contexte formel est généralement représenté par une matrice d'adjacence (n*m).

Exemple :

Dans cet exemple nous allons représenter un contexte formel reliant un ensemble d'objet O = {John, Maria, Peter, Clara} à un ensemble d'attributs P= {Homme, Femme, Mère, Père, Parent}, qui est illustré à travers la table [02] suivante :

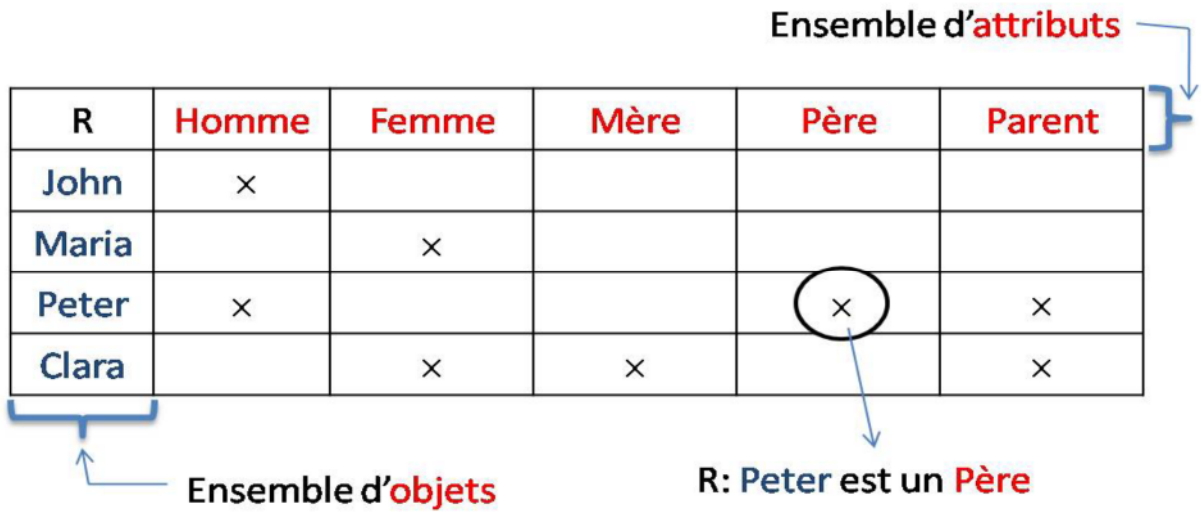


Table 1.2 : Contexte formel K.

Dans le contexte formel représenté par la table 1.2, l'objet Clara possède les attributs Femme, Mère et Parent.

I-5-2) Opérateur de dérivation de Galois :

Une grande partie du fond mathématique de l'analyse de données qualitatives concerne les opérateurs, les structures relationnelles et leur interaction. Parmi ces opérateurs on cite: l'opérateur de Galois.

En ACF, on définit des correspondances entre les ensembles 2^O et 2^P. Ces correspondances sont appelés opérateurs de dérivation de Galois, [03] proposé par Wille et noté $(.)^\Delta$.

Soit K = (O, P, R) un contexte formel, étant donnés deux ensembles $X \in 2^O$ et $A \in 2^P$, ces deux opérateurs sont définis comme suit :

X^{Δ} correspond aux attributs qui sont satisfaits par tous les objets de X:

$$X^{\Delta} = \{a \in P \mid \forall x \in O(x \in X \rightarrow xRa)\}$$
$$= \{a \in P \mid X \subseteq R(a)\}$$
$$= \cap_{x \in X} R(x)$$

A^{Δ} correspond aux objets qui satisfont tous les attributs de A :

$$A^{\Delta} = \{x \in O \mid \forall a \in P(a \in A \rightarrow xRa)\}$$
$$= \{x \in O \mid A \subseteq R(x)\}$$
$$= \cap_{a \in A} R(a)$$

I-5-3) Concept formel :

Soit K = (O, P, R) un contexte formel. Un concept formel noté ß (O, P, R) qui est un couple (X, A) dont $X \subseteq O$ $A \subseteq P$ tel que :

$$X^{\Delta} = A \text{ et } A^{\Delta} = X$$

X et A sont respectivement appelées extension (extent) et intension (intent) du concept formel (X, A).

Exemple :

Soit le contexte formel K (O, P, R), vu précédemment (table 02), $X \subseteq O$ et $A \subseteq P$

X^{Δ} = {Peter, Clara} $^{\Delta}$ = {Parent}. {Parent} est une intension.

A^{Δ} = {Parent} $^{\Delta}$ = {Peter, Clara}. {Peter, Clara} est une extension.

ß = < {{Peter, Clara}, {Parent}}> est appelé un concept formel.

Un concept formel fermé correspond à un rectangle maximal formé par les relations binaire du contexte dont tout objet de l'extension vérifie tous les attributs de l'intension ($A^{\Delta\Delta}$ = A et $X^{\Delta\Delta}$ = X).

L'ensemble de tous les concepts formels est noté par : $ß^{\Delta\Delta}$.

I-6)- Treillis de Galois :

Treillis de Galois ou treillis de concepts sont des structures clefs de l'ACF, c'est une forme de classification (ou plus précisément du clustering), plus élaborée qu'une simple partition ou une simple hiérarchie, qui peut ainsi être interprétée comme une relation de généralisation/ spécialisation entre les concepts formels. On appel treillis l'ensemble $(ß^{\Delta\Delta})$ de tous les concepts, muni de la relation d'ordre ($\leq$). [06]

Définition 4 :

Un treillis $(ß^{\Delta\Delta}, \leq)$ est un ensemble ordonné dans lequel deux éléments quelconques ont une borne supérieure (Sup) et une borne inférieure (Inf). Un treillis complet est un treillis pour lequel tout sous-ensemble possède une borne Sup et une borne Inf.

Nous allons donner un exemple de représentation graphique du treillis de Galois sous la forme d'un diagramme de Hasse, dans la figure 1.1 représentée ci-dessous, correspondant au contexte formel vu dans la table 1.2, où chaque rectangle représente un concept et les arcs entre les rectangles matérialisent la relation d'ordre du plus général (en haut) vers le plus spécifique (en bas).

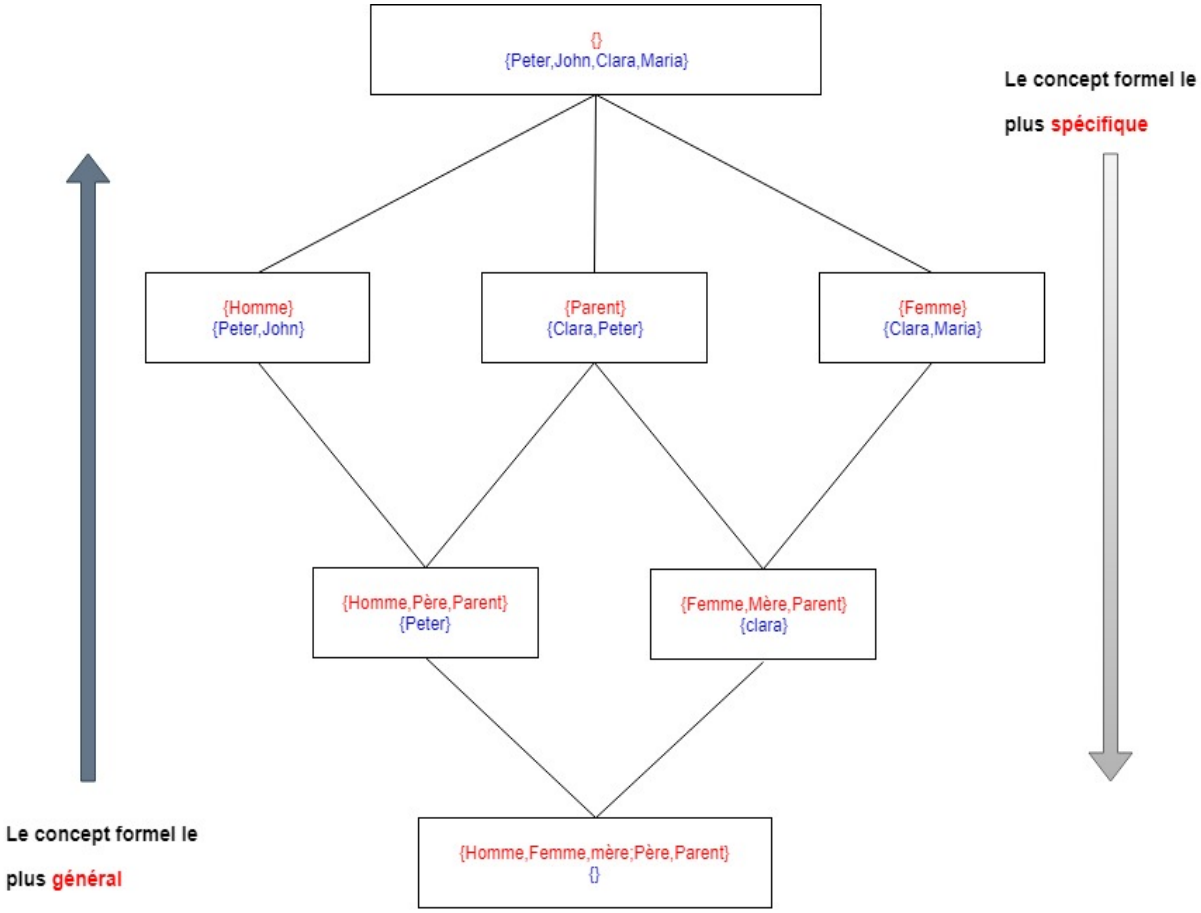


Figure 1.1 : Treillis de Galois.

La représentation graphique du treillis de concepts, sous la forme d'un diagramme de Hasse, facilite la compréhension et l'interprétation de la relation entre les objets et les attributs d'une part, et entre objets ou attributs d'autre part.

L'avantage de cette représentation est qu'à partir d'un treillis de concepts il est toujours possible de retrouver le contexte formel correspondant et inversement.

La construction du treillis de concepts d'une relation binaire donnée peut être décomposée en trois parties [07] :

- L'énumération des rectangles maximaux.
- La recherche de la relation d'ordre partiel entre ces rectangles.
- La représentation graphique du treillis (construction du diagramme de HASSE correspondant au treillis).

Plusieurs algorithmes de construction de treillis de Galois ont été proposé tel que : Chein [15], Norris [16], Ganter [17], NextClosure [17]…etc

Chacun de ces algorithmes se distingue des autres par plusieurs critères dont la stratégie de calcul des concepts, la recherche de l'ordre entre ces concepts.

I-7)- Implications d'attributs :

L'ACF s'est avérée être un moyen utile pour extraire des connaissances à partir des contextes formels sous forme d'implications d'attributs (conjonctives). Dans les définitions suivantes on trouve la condition nécessaire et suffisante pour qu'une implication d'attributs soit valide dans un contexte formel.

Définition 5 :

Soit un contexte formel K = (O, P, R) et $A1, A2 \subseteq P$ deux ensembles d'attributs. On dit que A1 implique A2 si et seulement tout objet de O qui a les attributs de A1 a aussi les attributs de A2.

$$A1 \rightarrow A2 \ ssi \ A1^{\Delta} \subseteq A2^{\Delta}$$

Exemple : Soit le contexte formel vu dans la table 1.1

$$\text{Mère} \rightarrow \text{Femme}$$

{ Clara}⊆ {Maria ,clara}d'où cette implication est dite valide .

Une implication de la forme $A1 \rightarrow A2$ tel que : (A1 est un ensemble d'attributs {a1, a2, …} ainsi A2), peut être ramenée à un ensemble d'implications de la forme $A1 \rightarrow$ ai $/$ i $=$ $\{1, 2, ...\}$ pour tout $a_i \in$ A2.

Exemple : Soit le contexte formel vu dans la table 1.1

L'attribut A1 {Mère} implique l'ensemble d'attributs A2 qui est {Femme et Parent}

$$\text{Mère} \rightarrow \text{Femme, Parent}$$

Peut être ramenée à un ensemble d'implications de la forme :

$$\text{Mère} \rightarrow \text{Femme}$$

ET

$$\text{Mère} \rightarrow \text{Parent}$$

Définition 6 : [05]

Etant donné un contexte K (O, P, R), une implication (entre attributs) $A1 \rightarrow A2$ avec $A1, A2 \subseteq P$ est valide dans K si et seulement si toute observation possédant tous les attributs de A1, possède aussi tous les attributs de A2 ($A1^{\Delta} \subseteq A2^{\Delta}$). L'ensemble A1 est alors appelé la prémisse et A2 la conclusion.

I-8)- Application de l'ACF :

A son apparition, l'ACF était essentiellement une théorie mathématique. Cependant, ses problématiques principales sont rapidement devenues de nature informatique [05]. Ce développement qui a permis son application dans une très grande variété de domaines tel que : Fouille de données, extraction des connaissances, construction d'ontologies [13], recherche d'informations…etc

I-8-1)-Application de l'ACF à la recherche d'information : [07]

La recherche d'information (RI) a été l'une des premières applications phare de treillis de concepts à la découverte de ressources. Les premiers travaux [09] [10] ont étudié la possibilité d'utiliser les treillis de concepts comme support pour la recherche documentaire. Des collections de documents sont alors représentées sous la forme de contextes formels. Les objets du contexte sont des documents et les attributs sont les termes d'indexation de ces documents. Chaque concept du treillis correspondant est vu comme un couple formé par une requête, dont les mots clés sont les termes contenus dans l'intension du concept, et l'ensemble de documents pertinents pour cette requête sont les documents contenus dans l'extension du concept.

Le calcul de la réponse à une requête donnée revient à identifier, dans le treillis, le concept dont l'intension est identique à la requête. Les liens de spécialisation/généralisation entre les concepts permettent d'effectuer une recherche progressive dans le treillis. Cette

façon de procéder suppose que la requête existe déjà dans le treillis. Pour assurer cette condition, des algorithmes de construction incrémentale de treillis de concepts sont utilisés pour l'insertion des requêtes dans un treillis déjà construit. De cette manière, un premier mode de recherche par treillis a été défini : la recherche par interrogation. La structure hiérarchique des treillis de concepts permet la définition d'un deuxième mode de recherche : la recherche par navigation.

Les deux modes de recherche, par interrogation et par navigation défini par l'utilisation des treillis de concepts, peuvent être combinés. Le treillis de concepts sert d'espace de recherche commun et assure la cohérence des résultats de recherche des deux modes.

I-8-1-1)-Recherche par interrogation :

Ce mode de recherche est facilité par la mise en place d'algorithmes performants pour la construction incrémentale des treillis de concepts. La définition de requête consiste à spécifier directement les termes d'indexation qui décrivent le(s) document(s) à trouver. La requête est ensuite insérée dans le treillis. La recherche des documents pertinents revient à localiser le concept le plus général incorporant les termes spécifiés dans la requête.

I-8-1-2)-Recherche par navigation :

La recherche par navigation utilise le treillis de concepts formels, consiste à explorer la structure hiérarchique de treillis visualisée pas un diagramme de Hasse. Ce dernier offre une interface de navigation permettant de se déplacer d'un concept formel à un autre.

On peut effectuer une navigation libre dans le treillis en suivant les liens descendants, partant du concept le plus général du treillis qui représente la classe de tous les documents avec un ensemble de termes communs souvent vide, on effectue une spécialisation graduelle en suivant les liens descendants dans le treillis. Chaque pas dans le treillis est équivalent à l'ajout d'un nombre minimal de nouveaux termes qui spécifient la description des documents à trouver. Cette opération est répétée jusqu'à l'identification du sous ensemble minimal de documents recherchés.

Dans d'autres cas, suivant les liens ascendants du treillis, partant d'une description très précise de documents, on peut relâcher progressivement cette description. Chaque étape correspond à la suppression d'un ensemble minimal de termes ce qui entraîne l'augmentation du nombre de documents qui satisfont la description allégée.

I-9)- Conclusion :

Après avoir expliqué quelques notions de bases sur l'analyse de concepts formels en présentant quelques exemples explicatifs de cette dernière, il s'avère que ce cadre est souvent trop restrictif pour représenter certaines réalités.

L'ACF est amenée à considérer des contextes de diverses natures, où apparaît le cas de l'ignorance qui est un cas où on ne sait pas si un objet possède un attribut ou pas, ce qui conduit à la notion de contexte formel incomplet que nous allons détailler dans le chapitre qui suit.

Chapitre II : Les implications dans un contexte formel incomplet

Sommaire :

II-1)-Introduction :

Classiquement, l'ACF s'applique à une relation binaire représentée par un contexte formel. Un objet satisfait un attribut ou ne le satisfait pas. Mais la réalité nous amène souvent à manipuler des informations imparfaites que les chercheurs ont classifiées en trois différentes catégories qui sont l'imprécision, l'incertitude et l'incomplétude.

L'imprécision des données consiste en informations indéterminées qui laissent place à une ambigüité, dans ce cadre les informations introduites ne sont pas claires. L'imprécision consiste à formaliser des quantificateurs tels que : « environ, grand, certaine…etc ». [04]

L'incertitude concerne la validité d'une information telle qu'on ne sait pas si elle est vraie ou fausse, par exemple en disant « Je pense qu'il pleuvra demain ».

L'incomplétude fait référence à une situation partiellement informée ou à une situation d'ignorance totale, ce qui nécessite de considérer des environnements incomplets où on tient compte de l'hypothèse du monde ouvert, Il est donc utile de voir ce que deviennent les connaissances obtenues à partir de ce contexte formel.

Nous présenterons dans ce qui suit, les cadres théoriques permettant de modéliser les informations incomplètes en définissant si une implication obtenue à partir de ce contexte est certaine ou pas.

Une deuxième contribution consiste à proposer une approche qui permet de générer toutes les implications certaines à partir d'un contexte formel incomplet.

II-2)- Contexte formel incomplet :

L'ACF est amenée à considérer des contextes de diverses natures dont les connaissances peuvent être incomplètes. Il est donc important 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.

Le cas de l'ignorance partielle dans des contextes a été considéré par Obiedkov [07] en utilisant la logique modale et par Burmeister et Holzer [08] en utilisant la logique à trois valeurs (Kleene 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.

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

L'interprétation de la relation **R** est comme suit. Soit $x \in O$ et $a \in P$:

_ $(x, a, +) \in R$: 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 ou ne vérifie pas l'attribut a.

Exemple : soit $K^{?}$ un contexte formel incomplet.

$K^{?}$	a_1	a_2	a_3
x_1	+	+	+
x_2	?	+	-
x_3	-	?	+

Table 2.1 : Contexte formel incomplet $K^{?}$

Un contexte formel incomplet peut être considéré comme la famille de tous les contextes formels standards obtenus en remplaçant les entrées inconnues (x, a, ?) par celles connues ((x, a, +) ou (x, a, -)).

Il existe exactement 2^n contextes formels obtenus en remplaçant chaque " ?" par "+" ou "-" (où n est le nombre de " ?" dans le contexte formel incomplet). Ils sont appelés contextes possibles.

Si K_1 et K_2 sont des contextes possibles obtenus à partir de $k^?$, on définit un ordre entre eux comme suit : $K_1 < K_2$ si $R_1 \subset R_2$ il ya plus de « + » dans R_2 que dans R_1. Ensuite il est facile de vérifier que :

Lemme0 1 : $K_1 < K_2$ implique que tout sous-ensemble A d'attributs, $A_{k_1}^{\Delta} \subseteq A_{k_2}^{\Delta}$

Preuve du Lemme 01 :

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 tous les attributs dans A pour K_2 que pour K_1.

Exemple: les contextes formels obtenus à partir du contexte formel incomplet $K^?$ donné dans la table 2.1 sont :

K_1	a_1	a_2	a_3
x_1	+	+	+
x_2	+	+	-
x_3	-	+	+

K_2	a_1	a_2	a_3
x_1	+	+	+
x_2	-	+	-
x_3	-	+	+

K_3	a_1	a_2	a_3
x_1	+	+	+
x_2	+	+	-
x_3	-	-	+

K_4	a_1	a_2	a_3
x_1	+	+	+
x_2	-	+	-
x_3	-	-	+

Table 2.2 : Tous les contextes formels possibles $k^?$

Considérant les deux cas extrêmes où toutes les entrées inconnues (x, a, ?) sont remplacées par (x, a, -), et le cas où toutes ces entrées inconnues (x , a, ?) sont remplacées par (x, a , +), cela donne naissance aux complétions inférieures et supérieures respectivement [04].

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

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

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

Exemple : dans le cas de la table 2.1 vue précédemment, les contextes formels k^* et k_* sont comme suit :

K^*	a_1	a_2	a_3
x_1	+	+	+
x_2	+	+	-
x_3	-	+	+

K_*	a_1	a_2	a_3
x_1	+	+	+
x_2	-	+	-
x_3	-	-	+

Table 2.3 : Contexte formel k^* et k_*

II-3)-Implications d'attributs conjonctives dans un contexte formel incomplet :

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 conjonctives certaines :

Notre étude s'est basée sur l'extraction des connaissances à partir d'un contexte formel incomplet, l'exploration de ces connaissances sous forme d'implications d'attributs conjonctives certaines qui tiennent dans tous les mondes possibles compatibles avec l'information incomplète.

***Définition 05*:**

Une implication d'attributs est dite certaine dans un contexte incomplet $K^?$ si et seulement si elle est valide dans chaque contexte formel K tel que : $K_* \leq K \leq K^*$.

Cette définition peut sembler difficile à vérifier en utilisant une énumération explicite des contextes formels possibles. Le théorème suivant fournit une solution à ce problème.

Théorème 01 : [20] $A \rightarrow B$ est une implication d'attributs certaine dans $K^?$ si et seulement si

$$A^{\Delta}_{K^*} \subseteq B^{\Delta}_{K_*}$$

Preuve du Théorème 01:

a) Condition nécessaire :

Soit $A \rightarrow B$ une implication d'attributs certaine dans $K^?$ et on suppose que :

$$A^{\Delta}_{K^*} \nsubseteq B^{\Delta}_{K_*}$$

$A^{\Delta}_{K^*} \nsubseteq B^{\Delta}_{K_*} \Longrightarrow \exists x \in O | \, x \in A^{\Delta}_{K^*} \; et \; x \notin B^{\Delta}_{K_*} \Longrightarrow \exists$ un contexte formel k_j (O, P, R_j) tel que :

$R_j = R_* \cup Q \; ou \; Q = \{(x, a) \in R^* | a \in A \wedge (x, a, ?) \in R\}$

ainsi, $x \in A^{\Delta}_{k_j} \; et \; x \notin B^{\Delta}_{K_j} \Longrightarrow A^{\Delta}_{k_j} \nsubseteq B^{\Delta}_{K_j} \Longrightarrow A \rightarrow B$ n'est pas valide dans $k_j \Longrightarrow A \rightarrow B$ n'est pas une implication d'attributs certaine.

b) Condition suffisante :

Si $A^{\Delta}_{K^*} \subseteq B^{\Delta}_{K_*}$ alors $A \rightarrow B$ est une implication d'attributs certaine dans $K^?$. En effet, pour tout contexte formel possible K_j il est valide que $A^{\Delta}_{K_j} \subseteq A^{\Delta}_{K^*} \subseteq B^{\Delta}_{K_*} \subseteq B^{\Delta}_{K_j}$. Par conséquent

A→B est une implication d'attributs certaine dans K_j. Pour tous les contextes formels possibles $K^?$.

Remarque : Une preuve directe de la condition nécessaire peut également être donnée.

Notez que A→ B est une implication d'attributs certaine si et seulement si A →b est une implication d'attributs certaine pour tout b ∈ B \ A. Dans ce qui suit nous pouvons se limiter aux implications d'attributs certaines A → b. Cela signifie que pour tout contexte *K* compatible avec $K^?$, $A_k^\Delta \subseteq \{b\}_k^\Delta$ est valide. Considérons l'ensemble des objets tels que $O_A = \{x \in O: \ a \in A, (x, a, -) \notin \boldsymbol{R}$. Ainsi, les lignes restreintes à A du contexte incomplet correspondant à O_A ne contiennent que "+" ou " ?". L'inclusion $A_k^\Delta \subseteq \{b\}_k^\Delta$ □ k compatible avec $K^?$ signifie que si (x ∈ O_A) alors (x; b; +) ∈ R. Donc nous pouvons voir que : $A_{K^*}^\Delta = O_A$, tandis que $\subseteq \{b\}_{K_*}^\Delta \supseteq O_A$.

II-3-2)-Implications d'attributs conjonctives possibles :

***Définition 06*:**

Une implication d'attributs est dite possible dans un contexte incomplet $K^?$ si et seulement si elle est valide au moins un contexte formel K tel que $K_* \leq K \leq K^*$.

Il est clair qu'une implication d'attributs certaine est également possible dans le sens de cette définition. Évidemment, l'inverse ne tient pas. Encore une fois, vérifier si une implication d'attributs est possible semble exiger une énumération des contextes possibles.

II-4)-Implications d'attributs disjonctives dans un contexte formel 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.

Afin de comprendre les implications d'attributs disjonctives, il est nécessaire de définir l'opérateur possibiliste Π utilisé dans les théorèmes.

II-4-1)-L'opérateur de possibilité $\prod$:

X^{Π} est l'ensemble des attributs satisfaits par au moins un objet dans X :

$$\begin{aligned} X^{\Pi} &= \{a \in P | \exists x \in X, aRx\} \\ &= \{a \in P | X \cap R(a) \neq \emptyset\} \\ &= \bigcup_{x \in X} R(x) \end{aligned}$$

A^{Π} est l'ensemble des objets qui satisfaits au moins un attribut dans A :

$$\begin{aligned} A^{\Pi} &= \{x \in O | \exists a \in A, xRa\} \\ &= \{x \in O | A \cap R(x) \neq \emptyset\} \\ &= \bigcup_{a \in A} R(a) \end{aligned}$$

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

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

II-4-2)-Implications d'attributs disjonctives certaines :

Théorème 02: [20] $\bigvee A \rightarrow \bigvee B$ est une implication d'attributs disjonctive certaine si et seulement si $A_{k^*}^{\Pi} \subseteq B_{k_*}^{\Pi}$

Preuve du Théorème 02:

D'une part, par le théorème 01, $B^{\neg} \rightarrow A^{\neg}$ est une implication d'attributs certaine dans $\overline{\boldsymbol{K}^{?}}$ si et seulement si $B_{\overline{K_*}}^{\Delta} \subseteq A_{\overline{K^*}}^{\Delta}$, puisque $\overline{\boldsymbol{K}_*}$ et $\overline{\boldsymbol{K}^*}$ sont respectivement la haute complétion et la basse complétion (the upper and the lower completions) dans $\overline{\boldsymbol{K}^{?}}$; cela signifie également que $\bigvee A \rightarrow \bigvee B$ est une implication d'attributs disjonctive certaine dans $K^{?}$.

D'autre part, nous avons : $B_{\overline{K_*}}^{\Delta} \subseteq A_{\overline{K^*}}^{\Delta}$ si et seulement si $A_{k^*}^{\Pi} \subseteq B_{k_*}^{\Pi}$. Il en découle que, $\bigvee A \rightarrow \bigvee B$ est une implication d'attributs disjonctive certaine si et seulement si $A_{k^*}^{\Pi} \subseteq B_{k_*}^{\Pi}$.

II-4-3)-Implications d'attributs disjonctives possibles :

Théorème 03 : [20] $\bigvee A \rightarrow \bigvee B$ est une implication d'attributs disjonctive possible si et seulement si $A_{k_*}^{\Pi} \subseteq B_{k^*{}_*}^{\Pi}$

Preuve du Théorème 03 :

D'une part, $B^{\neg} \rightarrow A^{\neg}$ est une implication d'attributs disjonctive possible si et seulement si $B_{\overline{K^*}}^{\Delta} \subseteq A_{\overline{K_*}}^{\Delta}$. Cela signifie également que $\bigvee A \rightarrow \bigvee B$ est une implication d'attributs disjonctive possible dans $K^{?}$.

D'autre part, $B_{\overline{K^*}}^{\Delta} \subseteq A_{\overline{K_*}}^{\Delta}$ est équivalent $A_{K_*}^{\Pi} \subseteq B_{K^*}^{\Pi}$. Par conséquent, nous obtenons que $\bigvee A \rightarrow \bigvee B$ est une implication d'attributs disjonctive possible si et seulement si $A_{K_*}^{\Pi} \subseteq B_{K^*}^{\Pi}$

II-5)-Toutes les implications d'attributs conjonctives certaines dans un contexte formel incomplet :

Après avoir introduit le théorème 01, qui vérifie la certitude d'une implication, notre contribution consiste à proposer une manière qui permet de générer toutes les implications

conjonctives certaines contenues dans un contexte formel incomplet, pour cela nous avons procédé comme suit :

Notons un contexte formel incomplet $K^?$ contenant n attributs, le nombre d'implications à vérifier est $(2^n - 1) \times (2^n - 1)$ (tel que : $(2^n - 1)$ est le nombre de combinaisons possibles pour chaque côté de l'implication)

Exemple : pour un contexte formel incomplet $K^?$ contenant 6 attributs (n=6) on aura : 3969 implications à vérifier.

D'autre part, citons que :

Pour toute implication $A \to B$ tel que : A et B sont deux ensembles d'attributs. Cette implication peut être ramenée à un ensemble d'implications de la forme $A \to \text{bi} \ / \ \text{i} = \{1, 2, \dots\}$ pour tout $b_i \in B$ ce qui nous a permis de garder un seul attribut en conclusion de l'implication et permuter les attributs de la prémisse donc le nombre d'implications à vérifier sera réduit à $(2^n - 1) \times n$.

Donc pour 6 attributs, utilisant cette dernière nous aurons seulement 378 implications à vérifier.

Enfin, pour dire qu'une implication est certaine :

On vérifier si les objets qui ne satisfont pas la conclusion dans K_*, ne satisfont pas aussi la prémisse dans K^*.

II-6)-Conclusion :

Après avoir pu modéliser le contexte formel en présence des informations incomplètes, l'étude s'est basée sur les deux extrémités (K^* , K_*) des contextes formels obtenus pour la vérification de la validité d'implications certaines en utilisant le théorème 01 ,ainsi une manière de générer toutes les implications certaines pouvant se trouver dans un contexte incomplet quiconque.

Chapitre II : Les implications dans un contexte formel incomplet

Dans le chapitre suivant, nous allons mettre en œuvre des algorithmes applicatifs afin de vérifier la certitude d'une implication et enfin pouvoir afficher toutes les implications certaines obtenues à partir d'un contexte formel incomplet.

Chapitre III : Analyse & Conception

Sommaire :

III-1)-Introduction :

Comme on l'a précisé précédemment l'ACF est amenée à traiter des contextes de divers nature dont le cas d'incomplétude vu dans le chapitre précédent, et afin de rendre facile son utilisation dans des applications.

Notre contribution consiste à utiliser le théorème 01 pour proposer des algorithmes applicatifs. Dans un premier temps nous allons implémenter une méthode qui permet de vérifier la certitude des implications introduites par l'utilisateur. Dans un deuxième temps de proposer et d'implémenter un algorithme qui génère toutes les implications d'attributs certaines d'un contexte formel incomplet donné par l'utilisateur.

III-2)- Codages des données

III-2-1)- Format d'introduction d'un contexte formel incomplet $K^?$:

La structure du fichier que nous avons utilisé dans le cas d'un contexte formel incomplet est une matrice représentée comme suit :

- La première ligne du fichier contient les attributs de ce contexte.

-La première colonne du fichier contient les objets du contexte formel.

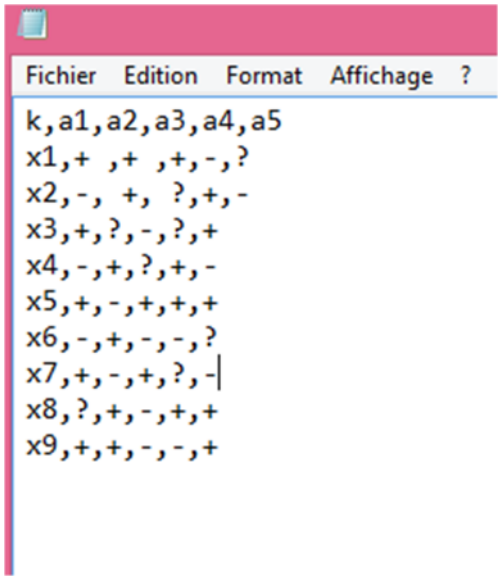
-L'intersection de la première ligne avec la première colonne représente le nom de ce contexte formel (K).

-L'intersection de la ligne avec la colonne représente la relation R= {+, -, ?} entre les objets et les attributs.

-Les lignes séparées par des sauts de lignes.

-Les colonnes sont séparées par des virgules.

La figure3.1. Représente une capture d'écran d'un fichier sur lequel nous avons sauvegardé un contexte formel incomplet K.



K	a1	a2	a3	a4	a5
x1	+	+	+	-	?
x2	-	+	?	+	-
x3	+	?	-	?	+
x4	-	+	?	+	-
x5	+	-	+	+	+
x6	-	+	-	-	?
x7	+	-	+	?	-
x8	?	+	-	+	+
x9	+	+	-	-	+

Figure 3.1 : Fichier représentant un contexte formel incomplet.

III-3)- Algorithme proposés :

Tout au long de notre étude, nous avons pu implémenter le théorème 01 présenté dans le chapitre précédent, afin de vérifier les implications d'attributs certaines dans un contexte formel incomplet puis une solution pour générer toutes les implications d'attributs certaines à partir d'un contexte formel incomplet.

III-3-1)- Lecture d'un contexte formel incomplet à partir d'un fichier :

Etant donné un contexte formel K, enregistré dans un fichier texte, sa lecture est effectuée par le programme suivant :

chemin_fichier : est une variable de type string, qui récupère le chemin de l'emplacement du fichier.

Ligne : récupère la ligne dans un fichier, est de type String.

Vecteur_ligne : est un vecteur de type String, de taille dynamique qui reçoit les lignes à partir du fichier.

Contexte_formel : est un tableau à deux dimensions de type String, de taille dynamique dans lequel on représente le contexte formel lu à partir d'un fichier.

Lecture d'un fichier :

```
Algorithme LireFichier :

Début
Ouvrir le fichier en lecture (chemin_fichier)
Tant que (non (EOF)) faire
   Début
        Lire la ligne suivante
        vecteur_ligne []← ligne
        pour (i allons de 1 à (longueur de (vecteur_ligne[]-1 )) faire
            Début
                Si  ( vecteur_ligne[i]  ≠ (-) ) alors
                    Contexte_formel [ligne_courante][i]← null ;
                Sinon
                    Contexte_formel[ligne_courante][i] ← vecteur_ligne[i]
                FinSi
              Fin
   Fin
Fin
```

III-3-2)-Génération du contexte formel K^*:

Après avoir lu le contexte formel incomplet enregistré déjà dans un fichier, on donne l'algorithme qui génère le contexte formel booléen K^* dont on remplace toutes les entrées «?» par « + ».

K_plus : est un tableau à deux dimensions de type String, de la même taille que le contexte formel, où on a remplacé toutes les relations de types { ?} par {+}, (K_plus représente K^*).

Contexte Formel K^* :

```
Algorithme  K_plus :
Début
   LireFichier()
      Pour ( i allons de 0 à( nbr_de_ligne-1) ) faire
      Début
         Pour ( j allons de 0 à (nbr_de_colonne – 1) ) faire
          Début
            Si ( Contexte_formel[i][j] ="? ") alors
              K_plus[i][j] ← "+"
            Sinon
              K_plus[i][j] ← Contexte_formel[i][j]
            FinSi
         Fin
       Fin
Fin
```

III-3-3)-Génération du contexte formel K∗:

Après avoir lu le contexte formel incomplet enregistré déjà dans un fichier, on donne l'algorithme qui génère le contexte formel booléen K∗ dont on remplace toutes les entrées «?» par « - ».

K_moins : est un tableau à deux dimensions de type String, de même taille que le contexte formel, où on a remplacé toutes les relations de types { ?} par {-}, (K_moins représente K∗).

Contexte Formel K_* :

```
Algorithme  K_moins :
Début
   LireFichier()
      Pour ( i allons de 0 à( nbr_de_ligne-1) )faire
      Début
         Pour ( j allons de 0 à (nbr_de_colonne – 1) )faire

          Début
            Si ( Contexte_formel[i][j] ="? ") alors
              K_moins[i][j] ← "-"
            Sinon
              K_moins[i][j] ← Contexte_formel[i][j]
            finSi
         Fin
       Fin
Fin
```

III-3-4)-Vérification d'implications d'attributs certaines :

Dans cette phase, on a pu proposer un algorithme qui vérifie le théorème 01, vu dans le chapitre précédent tel que cet algorithme permet de vérifier la certitude d'une implication saisie par un utilisateur pour le contexte formel incomplet introduit dans le fichier.

Premisse : est un vecteur de type String, où on récupère les attributs prémisses saisie par un utilisateur.

Conclusion : est un vecteur de type String, où on récupère les attributs conclusion saisie par un utilisateur.

Objet_premisse : est un vecteur de type String, où on récupère les objets qui satisfont les attributs du vecteur premisse dans K^*.

Objet_conclusion : est un vecteur de type String, où on récupère les objets qui satisfont les attributs du vecteur Conclusion dans K∗.

- Récupérer tous les objets du contexte formel K^* vérifiant toutes les prémisses en même temps dans un vecteur (objet_premisse).

Implications certaines :

```
Algorithme Implications certaines :
Début
   LireFichier ()
   K_plus ()
   K_moins ()

      Pour ( i allons de 0 à( nbr_de_ligne-1) )faire
      Début
        Pour ( k allons de 0 à( longueur Premisse - 1) ) faire
         Début
         Pour ( j allons de 0 à (nbr_de_colonne – 1) )faire
          Début
   Si (Premisse [k]= K_plus [0][j])alors
   Si (K_plus  [i][j]≠ null)alors
      P++
      FinSi
      FinSi
      Fin
      Fin
      Si(p = longueur Premisse) alors
      Pour (l allons de 0 à( longueur Objet_premisse - 1) ) faire
            l++
         Objet_premisse[l] ← K_plus [i][0]
```

```
        Fin
        FinSi
        p←0
    Fin
```

❖ Récupérer tous les objets du contexte formel K_* vérifiant toutes les conclusions en même temps dans un vecteur (Objet_conclusion).

Implications certaines :

Algorithme Implications certaines :(suite)

```
        Pour ( i allons de 0 à( nbr_de_ligne-1) )faire
        Début
          Pour ( k allons de 0 à( longueur Conclusion - 1) ) faire
           Début
           Pour ( j allons de 0 à (nbr_de_colonne – 1) )faire
            Début
        Si (Conclusion [k]= k_moins[0][j]) alors
        Si (k_moins[i][j]≠ null) alors
        c++
        FinSi
        FinSi
        Fin
        Fin
        Si ( c = longueur Conclusion) alors
        Pour (l allons de 0 à ( longueur Objet_conclusion - 1) ) faire
                l++
           Objet_conclusion[l] ← k_moins[i][0]

          Fin
          FinSi
```

```
c←0
Fin
```

❖ Vérifier si tout objet récupéré dans (Objet_premisse) est inclus dans (Objet_conclusion), pour dire que l'implication est certaine.

Implications certaines :

```
Algorithme Implications certaines :(suite)

    Pour ( i allons de 0 à( longueur Objet_premisse -1) )faire
    Début
      Pour ( j allons de 0 à( longueur Objet_conclusion - 1) ) faire
       Début
    Si (Objet_premisse[i]= Objet_conclusion[j]) alors
    Cpt++
    FinSi

      Fin
      Fin
      si (cpt= longueur Objet_premisse) alors
      Ecrire ("l'implication est certaine")
      sinon
      Ecrire ("l'implication est incertaine")
    Fin Si
```

III-3-5)-Génération de toutes les implications d'attributs certaines:

Le rôle de cet algorithme est la génération de toutes les implications certaines à partir d'un contexte formel et pour cela nous avons procéder comme suit :

- Récupérer dans un vecteur les objets qui ne satisfont pas l'attribut (conclusion) dans K_*.
- Vérifier dans le contexte formel K*, pour chaque combinaison d'attributs si les objets du vecteur ne la satisfont pas.
- Afficher toutes les implications vérifiées dans une fenêtre à l'utilisateur.

Avant de décrire l'algorithme général, nous allons d'abord définir les variables utilisées :

Vecteur : est un vecteur de type entier, où on récupère les indices des objets qui ne satisfont pas l'attribut (conclusion) dans K_*.

Attribut : est une variable de type string, qui récupère l'attribut conclusion à chaque fois.

Implication: est une variable de type string, où on récupère les implications vérifiées.

Toutes les implications certaines :

```
Algorithme Implications certaines :
   LireFichier ()
   K_plus ()
   K_moins ()
Début
// Récupèrer les objets qui ne satisfont pas l'attribut (conclusion) dans K*.

      Pour ( i allons de 0 à(nbr_de_colonne -1) )faire
      Début
     Cpt=0 ;
     Attribut←k_moins[0][j1]

     Pour (l allons de 1 à (nbr_de_ligne -1) faire
      Début
     Si (k_moins[l][j1]=null) alors
```

```
Vecteur [k1] ←l
K1++
Cpt ←k1
Fin Si
Fin

//cas ou tous les objets sont en relation avec l'attribut

Si (cpt=0) alors
Pour (i1 allons de 1 à nbr_de_colonne-1)faire
  Début
  Si (attribut≠ k_plus[0][i1]) alors
Implication ← implication +k_plus[0][i1]+ "-->"+attribut+"\n"
Fin Si
Fin
Fin Si

// Vérifier dans le contexte formel K*, pour chaque combinaison d'attributs si les objets
du vecteur  ne la satisfont pas.

Pour (i1allons de 1à nbr_de_colonne -1) faire
K1←0
  Si (attribut≠ k_plus[0][i1]) alors
Si (k_plus [vecteur[k1]][i1]=null) alors
K1++
Si (k1=cpt) alors
Implication ← implication +k_plus[0][i1]+ "-->"+attribut+"\n"
Fin Si
Fin Si
Fin Si
Fin
Fin

//on procède de la même manière pour les prémisses à plusieurs attributs
```

III-4)- Organigramme de l'application :

Un organigramme est une représentation schématique des liens et des relations qui existent entre les procédures implémentées dans les algorithmes cités.

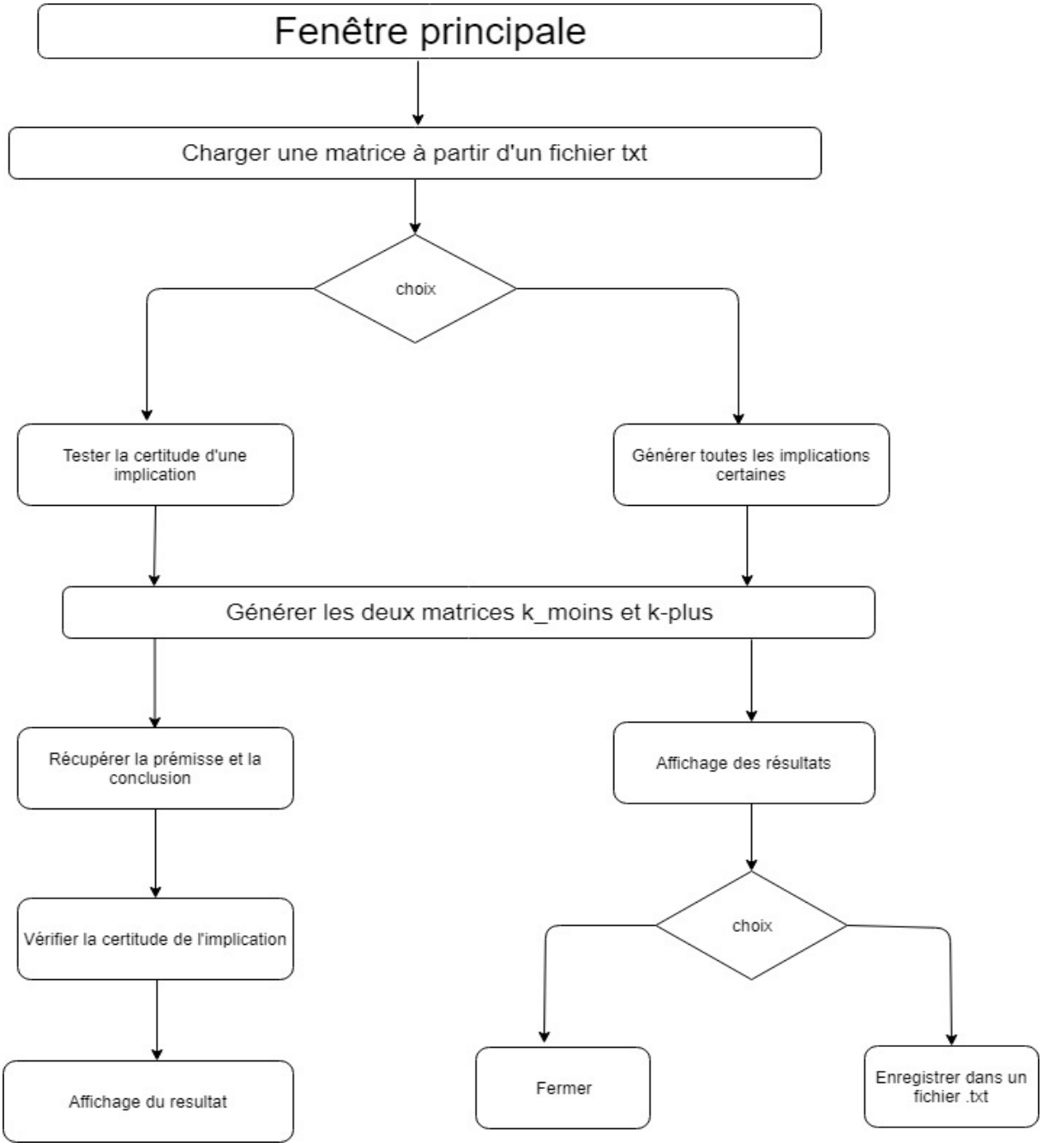


Figure 3.2: Organigramme représentant l'application.

III-5)- Conclusion :

Dans ce chapitre nous avons pu mettre en œuvre un algorithme qui vérifie les implications d'attributs certaines dans un contexte formel incomplet puis un autre algorithme qui génère toutes les implications d'attributs certaines de ce contexte formel incomplet.

Dans le chapitre qui suit, nous allons présenter une application illustrative des algorithmes proposés afin de faciliter leurs utilisations.

Chapitre IV : Réalisation & application

Sommaire :

IV-1)-Introduction :

Après avoir finalisé l'étape d'analyse et conception, nous passons à la phase réalisation qui consiste à implémenter les différents algorithmes vus précédemment dans une interface graphique afin de mieux comprendre leurs l'utilité.

Dans ce chapitre, nous allons présenter l'environnement de développement et les outils qui ont servi à la réalisation de notre application, ensuite nous présenterons la façon dont elle fonctionne avec ses différentes interfaces.

IV-2)- Description de l'environnement de développement :

IV-2-1)-Langage utilisé :

JAVA : C'est un langage de programmation orienté objet, développé par Sun Microsystems. Présenté officiellement le 23 Mai 1995 au SunWorld qui a été ensuite rachetée en 2009 par la société Oracle qui détient et maintient désormais Java. Une de ses plus grandes forces est son excellente portabilité : une fois votre programme créé, il fonctionnera automatiquement sous Windows, Mac, Linux, …etc

L'un des principes phares de Java réside dans sa machine virtuelle JVM (Java Virtual Machine): celle-ci assure à tous les développeurs Java qu'un programme sera utilisable avec tous les systèmes d'exploitation sur lesquels est installée une machine virtuelle Java. Lors de la phase de compilation de notre code source, celui-ci prend une forme intermédiaire appelée byte code : c'est le fameux code inintelligible pour votre machine, mais interprétable par la machine virtuelle Java. Cette dernière porte un nom : on parle plus communément de JRE (Java Runtime Environment). Plus besoin de se soucier des spécificités liées à tel ou tel OS (*Operating System*, soit système d'exploitation).

Citons aussi JDK (Java Development Kit), qui est l'ensemble des outils dont le développeur a besoin pour développer des logiciels basés sur Java.

Le JRE contient tout le nécessaire pour que vos programmes Java puissent être exécutés sur votre ordinateur ; le JDK, en plus de contenir le JRE, contient tout le nécessaire pour développer, compiler…

IV-2-2)-Outils de développement :

Eclipse IDE (**I**ntegrated **D**evelopment **E**nvironment) : est un environnement de développement libre permettant de créer des programmes dans de nombreux langages de programmation (Java, C++, PHP…). C'est l'outil que nous allons utiliser pour programmer.

Dans notre étude, nous avons installé (JRE, JDK, Eclipse IDE 12-2018) pour pouvoir ensuite commencer la programmation.

IV-3)-Présentation des interfaces de l'application :

L'interface graphique a été développée pour permettre à l'utilisateur de tester la validité d'une implication, il pourra aussi générer toutes les implications certaines d'un contexte formel donné.

Dans un premier lieu, l'interface principale qui apparaît à l'utilisateur est la suivante :

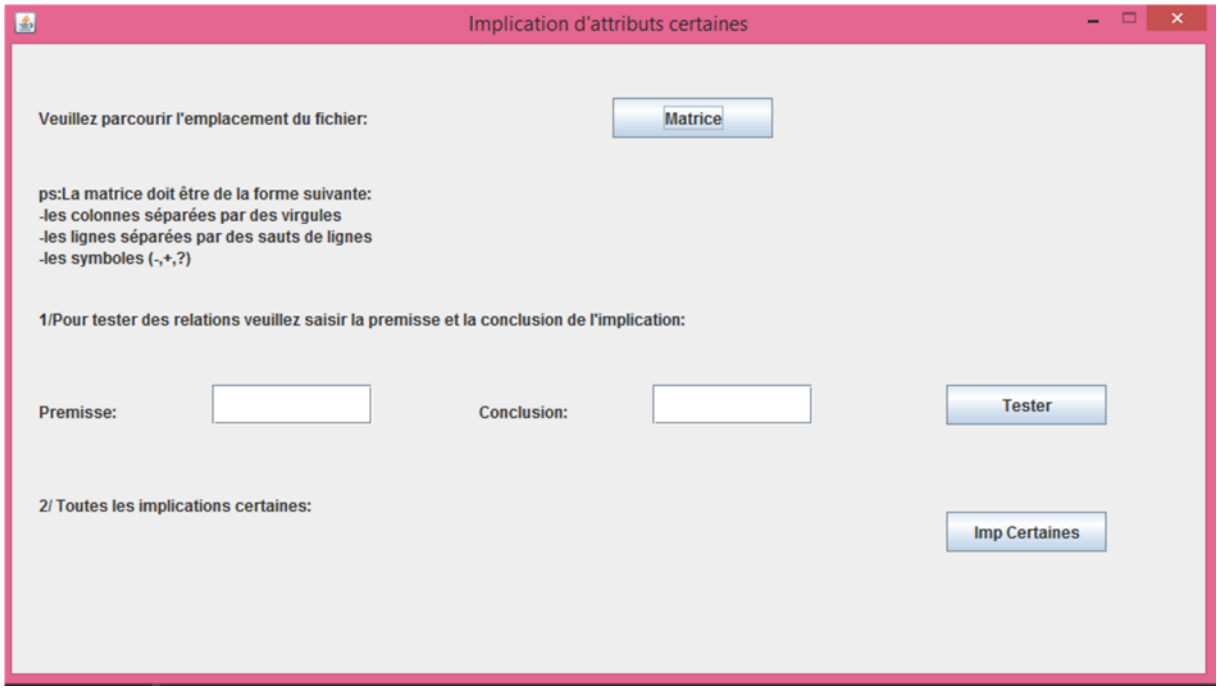


Figure 4.1 : Interface de l'application.

L'interface est simple car elle propose à l'utilisateur 3 boutons et deux champs à remplir :

- Matrice (pour parcourir un fichier)
- Tester (pour vérifier la certitude d'une implication) après avoir remplis les deux champs (Premisse et Conclusion)
- Imp Certaines (pour pouvoir générer toutes les implications d'attributs certaines).

IV-3-1)- Parcourir le contexte formel :

Une fois l'utilisateur s'est rendu sur la première interface, il peut parcourir son contexte formel incomplet K [?] qui a été déjà enregistré dans un fichier texte en cliquant sur le bouton Matrice (on pourra charger le fichier qu'on veut)

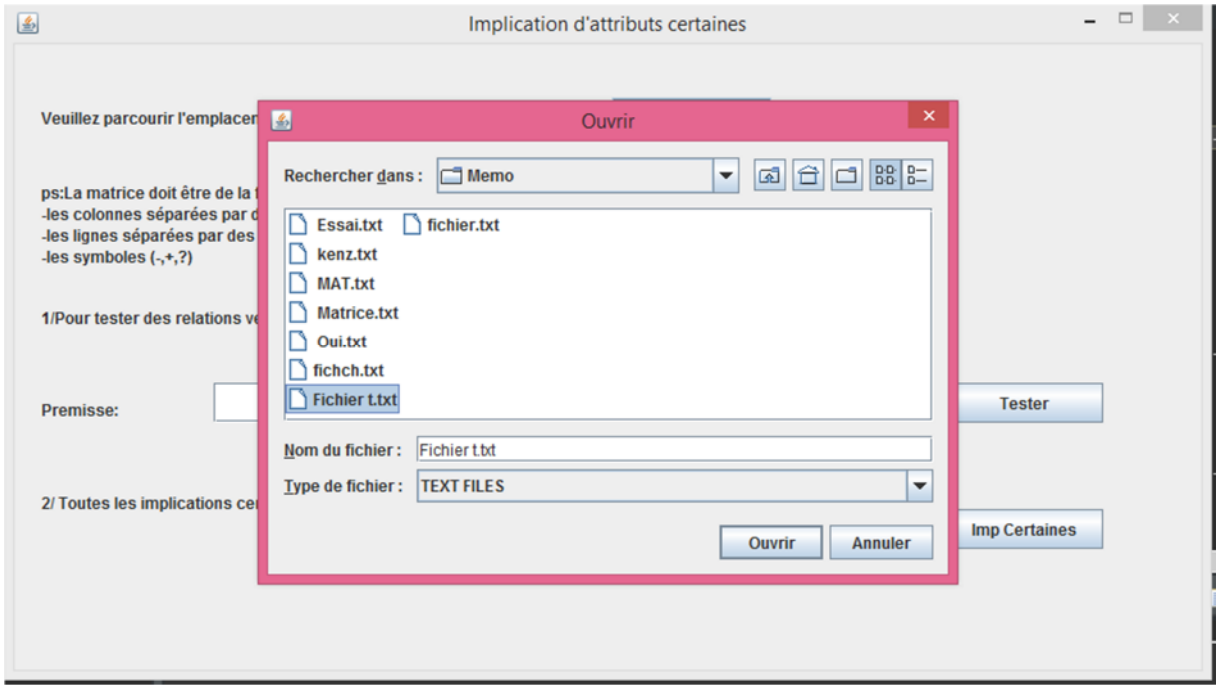


Figure 4.2 : Parcourir l'emplacement du fichier.

IV-3-2)- Tester la certitude des implications :

Cette partie de la fenêtre permet la vérification de la certitude d'une implication d'attribut saisie par l'utilisateur, en se basant sur l'algorithme d'implications d'attributs certaines vu dans le chapitre précédent, la capture suivante permet d'illustrer ce dernier.

Figure 4.3 : Vérification des implications certaines.

Pour vérifier une implication si elle est certaine ou pas, l'utilisateur doit remplir deux champs tels que la prémisse de l'implication dans un champ où les attributs doivent être séparés par des virgules, ainsi que la conclusion dans l'autre champ.

Exemple explicatif :

Pour un contexte formel déjà donné, on teste les implications suivantes :

01) $a1, a2, a3 \rightarrow a5$

Cette implication est représentée dans l'interface comme suit :

Figure 4.4 : Tester une implication.

En cliquant sur le bouton Tester, un message nous sera affiché pour dire que cette dernière $(a1, a2, a3 \rightarrow a5)$ est une implication incertaine.

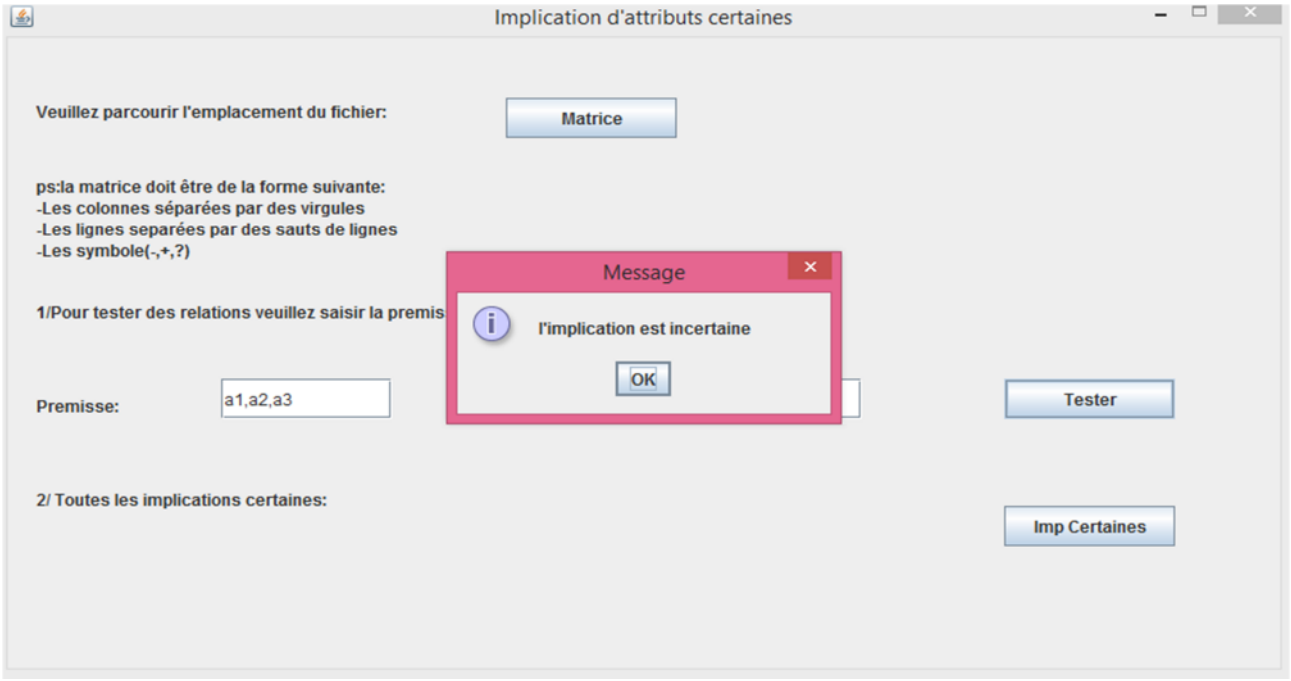


Figure 4.5 : Vérification de l'implication 01.

Une autre implication à tester :

02) $a1, a3 \rightarrow a2$

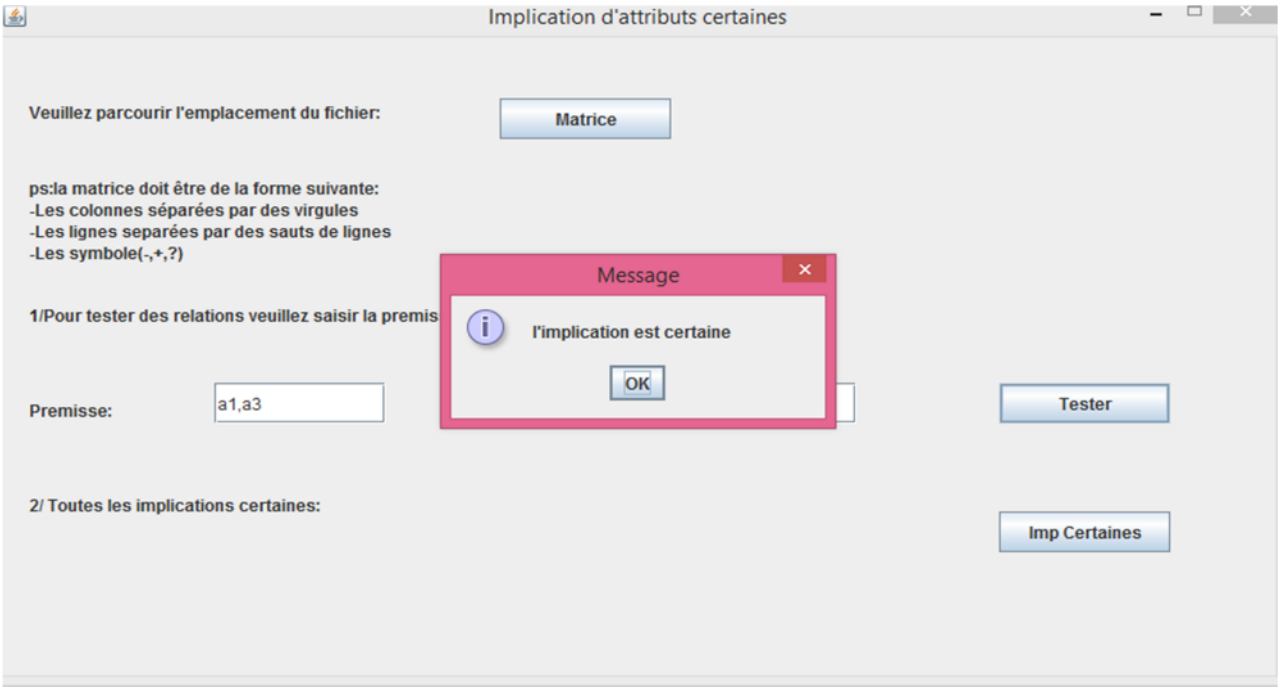


Figure 4.6 : Vérification de l'implication 02.

Le message nous sera affiché pour dire que cette implication $(a1, a3 \rightarrow a2)$ est une implication certaine.

IV-3-2)- Génération de toutes les implications certaines:

Cette partie permet à l'utilisateur de générer toutes les implications certaines en cliquant sur le bouton (Imp Certaines).

Une nouvelle fenêtre sera affichée. L'utilisateur pourra visualiser toutes les implications certaines du contexte formel choisi, et à la fin il y'a deux boutons qui sont proposés à l'utilisateur (Enregistrer et Fermer).

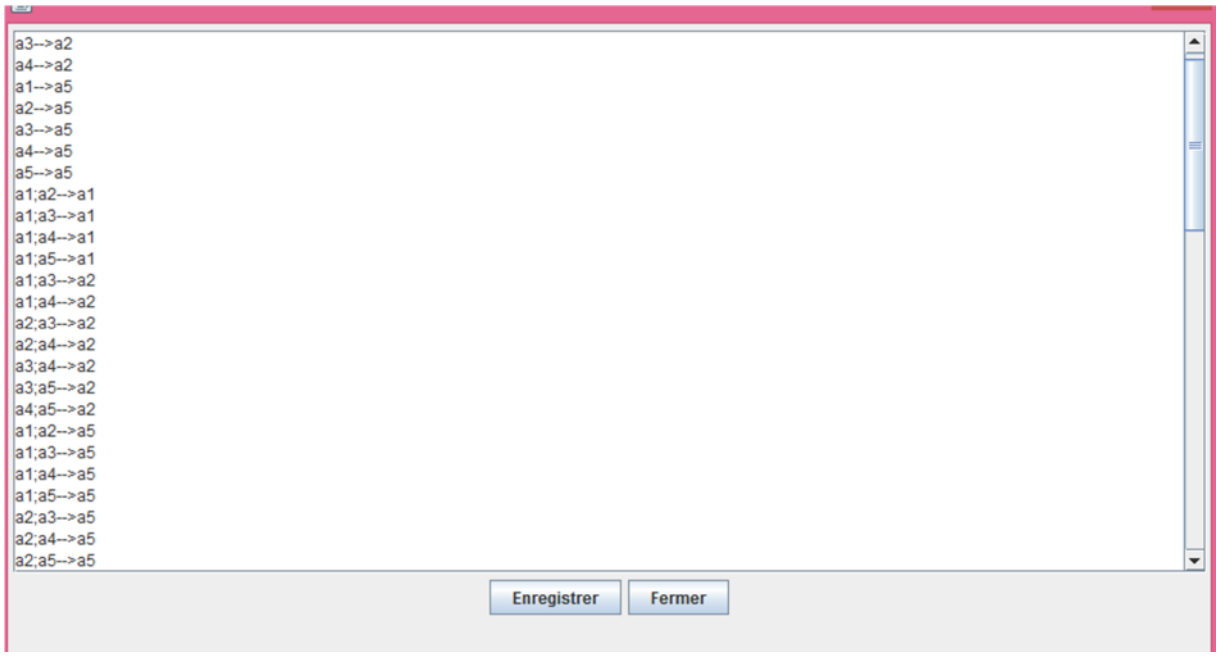


Figure 4.7: Fenêtre d'affichage des implications certaines.

Le bouton Enregistrer permet de sauvegarder ces implications dans un fichier texte et parcourir son emplacement.

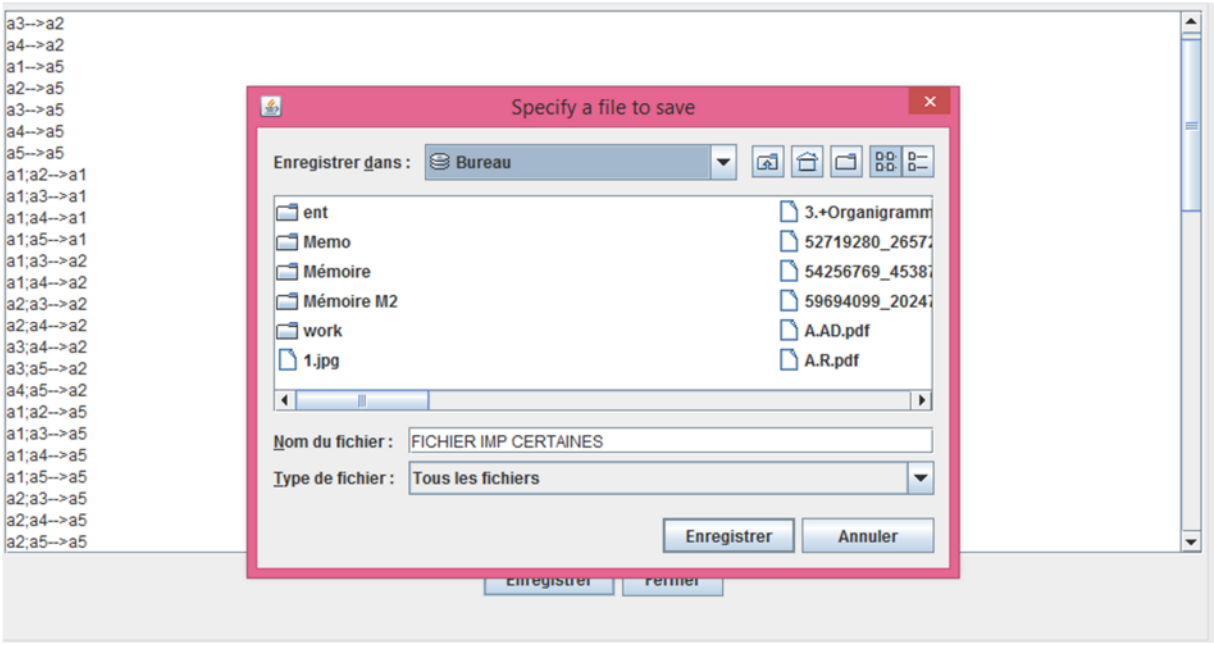


Figure 4.8 : Enregistrement des implications certaines.

IV-4)- Conclusion :

Dans ce dernier chapitre, nous avons présenté l'environnement d'implémentation et de développement de notre application, en se focalisant sur les techniques de programmation utilisées.

Ensuite nous avons expliqué son fonctionnement en présentant quelques interfaces.

Conclusion et perspectives :

Dans le cadre de ce mémoire, nous avons introduit l'analyse de concepts formels (ACF) en général, puis on a pris en considération le cas d'un contexte formel incomplet où nous nous sommes intéressées aux implications d'attributs certaines résultantes à partir de ce type de contexte.

On a pu implémenter des algorithmes pour ce cas d'extraction d'implications d'attributs certaines où on a proposé un algorithme, qui se base sur le théorème 01, qui vérifie si une implication introduite par un utilisateur est certaine ou pas. Nous avons aussi mis en œuvre un autre algorithme qui génère toutes les implications d'attributs certaines obtenues à partir d'un contexte formel incomplet.

Le sujet abordé dans ce mémoire ouvre diverses perspectives, nous en présentons quelques unes :

- ✓ Pour générer toutes les implications d'attributs certaines, le nombre de combinaisons d'attributs possibles ($(2^n -1) \times n$) semble énorme ce qui engendre l'explosion combinatoire dont la complexité de l'algorithme est de type exponentielle, mais ce problème peut être contourné à l'aide des outils mathématiques et algorithmiques. [18]

- ✓ Dans notre travail, La base d'implication générée n'est pas minimale, ce qui ouvre la voie à de nouvelles recherches.

Bibliographie :

[01] Marianne Huchard. Analyse Formelle de Concepts : Une approche pour fouiller des ensembles de données multi-relationnels, et quelques applications au génie logiciel. 2012.

[02] Zina Ait Yakoub Yassine Djouadi, UMMTO, Equipe LCD, ait.yakoub@hotmail.fr, USTHB, Equipe LCD djouadi@usthb.dz. JOURNEES SCIENTIFIQUES DU LABORATOIRE RIIMA, 8 9 Novembre 2015

[03] Yassine Djouadi. Généralisation des opérateurs de dérivation de Galois en recherche d'information basée sur l'analyse formelle de concepts Université de Tizi-Ouzou, BP 17, 15000 Tizi-Ouzou, Algérie. IRIT de Toulouse, France. Université Paul Sabatier, 118 Route de Narbonne, 31062 Toulouse Cedex 09, France djouadi@irit.fr

[04] Zina AIT YAKOUB. Logiques de Description & Analyse de Concepts Formels pour des représentations incomplètes. Thèse de doctorat LMD, discipline : informatique, option « Intelligence Artificielle et Systèmes d'Information ».

[05] Jean-François Viaud. Cadre général pour la recherche d'information et l'extraction de connaissances par l'exploration de treillis.

[06] Engelbert MEPHU NGUIFO — Patrick NJIWOUA. Treillis de concepts et classification supervisée

[07] Nizar Messai Analyse de concepts formels guidée par des connaissances de domaine : Application à la déccouverte de ressources génomiques sur le Web, le 20 Mars 2009 à l'université Henri Poincaré – Nancy 1 (spécialité informatique).

[08] R. Bendaoud, A. Napoli, and Y. Toussaint. Formal concept analysis : A unified framework for building and refining ontologies. In EKAW, 2008.

[09] C. Carpineto and G. Romano. Galois: An order-theoretic approach to conceptual clustering. In Proceedings of the 10th International Conference on Machine Learning (ICML'90), July 1993.

[10] Robert Godin, Rokia Missaoui, and Alain April. Experimental comparison of navigation in a galois lattice with conventional information retrieval methods. International Journal of Man-Machine Studies, 1993.

[11] Rudolf Wille. Formal concept analysis as mathematical theory of concepts and concept hierarchies. In Bernhard Ganter, Gerd Stumme, and Rudolf Wille, editors, Formal Concept Analysis, volume 3626 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2005.

[12] Birkhoff, G (1967). Lattice Theory, volume 25 of ASM Colloquium Publications. AMS, Providence, RI, 3rd edition. 1st ed., 1940 ; 2nd ed., 1948.

[13] Thibault Mondary, Sylvie Després. Analyse de concepts formels pour la construction d'ontologies à partir de textes : la question du corpus.

[14] B. Ganter and R. Wille. Formal Concept Analysis : Mathematical Foundations. Springer-Verlag, Berlin, Heidelberg, 1999.

[15] M. Chein. Algorithme de recherche des sous-matrices premières d'une matrice. In Bull. Math. Soc. Sci. Math. R.S. Roumanie, 1969.

[16] E. M. Norris. An algorithm for computing the maximal rectangles in a binary relation. In Revue Roumaine de Mathématiques Pures et Appliquées, 1978.

[17] B. Ganter. Two basic algorithms in concept analysis. FB4–Preprint 831, TH Darmstadt, 1984.

[18] Pierre Fouilhoux Université Pierre et Marie Curie. Contourner l'explosion combinatoire, Séminaire de vulgarisation Normandie-Mathématiques, 27 janvier 2016.

[19] Abdoulkader OSMAN GUEDI. Évolution et Transformation automatisée de modèles de Systèmes d'Information Une approche guidée par l'Analyse Formelle de Concepts et l'Analyse Relationnelle de Concepts.

[20] Zina Ait-Yakoub, Yassine Djouadi, Didier Dubois, and Henri Prade. Asym-metric composition of possibilistic operators in formal concept analysis. ap-plication to the extraction of attribute implications from incomplete contexts. International Journal of Intelligent Systems, 32(12) :1285-1311, 2017.

[21] Y. Djouadi, D. Dubois, and H. Prade. Possibility theory and formal concept analysis: Context decomposition and uncertainty handling. In Computational Intelligence for Knowledge-Based Systems Design, springer, 2010.

[22] Yassine Djouadi, Didier Dubois, and Henri Prade. On the possible meanings of degrees when making formal concept analysis fuzzy (regular paper). In Pedro Burillo, Humberto Bustince, Bernard De Baets, and János Fodor, editors, EUROFUSE Workshop Preference Modelling and Decision Analysis, Pampelune (Espagne), 16/09/2009-18/09/2009, pages 253–258, http ://www.unavarra.es/, 2009. Universidad Pública de Navarra.

[23] Zina Ait-Yakoub, Yassine Djouadi, Didier Dubois, and Henri Prade. Concepts formels incomplets ou incertains : vers une généralisation possibiliste de l'ana-lyse formelle de concepts (regular paper). In Rencontres Francophones sur la Logique Floue et ses Applications (LFA), La Rochelle, 02/11/2016-04/11/2016, pages 203-210, http://www.cepadues.com/, 2016. Cépaduès Editions.

[24] Z. Ait-Yakoub, Y. Djouadi, D. Dubois, and H. Prade. From a possibility theory view of formal concept analysis to the possibilistic handling of incomplete and uncertain contexts. In CEUR Proc. Workshop FCA for AI, The Hague, 2016.