République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université M'hamed BOUGARA de BOUMERDES

Faculté des Sciences

Département d'Informatique

MEMOIRE DE MAGISTER

Spécialité : Système informatique et génie des logiciels

Option : Spécification de Logiciel et Traitement de l'Information

Ecole Doctorale

Présenté par :

Amroun Karima

Thème

Découverte des dépendances fonctionnelles floues dans des modèles relationnels sous imprécision.

Devant le jury de soutenance composé de:

Mr. MEZGHICHE Mohamed	Professeur	UMBB	Président.
Mr. DJOUADI Yassine	Maître de conférence	UMMTO	Rapporteur.
Mme. AISSANI-MOKHTARI Aicha	Professeur	USTHB	Examinatrice.
Mme. ALI MAZIGHI Zaïa	Professeur	USTHB	Examinatrice.

Année Universitaire : 2007/2008

REMERCIEMENTS

Je tiens d'abord à remercier Dieu de m'avoir donné courage et santé pour accomplir ce travail.

Mes vifs remerciements accompagnés de toute ma gratitude vont ensuite à mon encadreur monsieur DJOUADI Yassine-Mansour, maître de conférence à l'UMMTO, pour m'avoir accueillie dans son équipe de recherche, pour ces remarques et conseils judicieux qui ont beaucoup contribué à l'amélioration de ce mémoire et surtout, pour sa rigueur scientifique. Je le remercie également pour m'avoir initiée à la recherche et de m'avoir fait bénéficier de ces connaissances.

Je voudrais également remercier vivement monsieur le président du jury MEZGHICHE Mohamed, professeur et directeur de l'école doctorale d'Informatique à l'UMBB, pour nous avoir accueilli dans son laboratoire LIFAB, et nous avoir fourni une formation riche pendant notre première année de magister. Puisse-t-il trouver ici l'expression de mon profond respect et de ma reconnaissance.

Je tiens à remercier vivement madame ALI MAZIGHI Zahia, professeur et doyenne de la faculté d'Informatique et d'Electronique à l'USTHB, qui nous a fait l'honneur d'accepter de juger ce modeste travail et y avoir consacré son précieux temps.

De la même manière, je tiens à remercier vivement madame AISSANI-MOKHTARI Aicha, professeur à l'USTHB, qui nous a fait l'honneur d'accepter de juger ce modeste travail et y avoir consacré son précieux temps.

Je tiens à remercier également le professeur BELKAID Mohand Said, doyen de la faculté de Génie Electrique et Informatique à l'UMMTO, pour toute l'aide qui nous a offerte. Puisse-t-il trouver ici l'expression de mes sincères respects.

Enfin, que tous ceux qui nous ont aidés et encouragés, de près ou de loin, retrouvent notre gratitude et sincères remerciements.

Table des matières

Chapitre III : Extraction de Connaissances, Data Mining et Inférence des Dépendances Fonctionnelles

Chapitre IV : Théorie des ensembles flous, bases de données et dépendances fonctionnelles floues

Chapitre V : Contributions

Conclusion et perspectives

Annexe 1 : Fermeture et treillis

Annexe 2 : correction et complétude du système de règles d'inférence des α-FFDs

Liste des définitions

Chapitre II : Les bases de données relationnelles

Chapitre III : Extraction de Connaissances, Data Mining et Inférence des Dépendances Fonctionnelles

Chapitre IV : Théorie des ensembles flous, bases de données et dépendances fonctionnelles floues

Chapitre V : Contributions

Annexe 1 : Fermeture et treillis

Annexe 2 : correction et complétude du système de règles d'inférence des α-FFDs

Table des figures

Chapitre III : Extraction de Connaissances, Data Mining et Inférence des Dépendances Fonctionnelles

Chapitre IV : Théorie des ensembles flous, bases de données et dépendances fonctionnelles floues

Annexe 1

Liste des algorithmes

Chapitre III : Extraction de Connaissances, Data Mining et Inférence des Dépendances Fonctionnelles

Chapitre IV : Théorie des ensembles flous, bases de données et dépendances fonctionnelles floues

Chapitre V : Contributions

Annexe 2 : correction et complétude du système de règles d'inférence des α-FFDs

CHAPITRE 1

INTRODUCTION GÉNÉRALE

Ce chapitre est destiné en premier lieu à présenter le problème auquel nous nous intéressons, c'est-à-dire « la découverte des dépendances fonctionnelles floues dans des modèles relationnels sous imprécision ». Tout d'abord, nous exposons les motivations qui nous ont conduit à nous intéresser à ce problème. Nous présentons ensuite, les contributions apportées par ce travail. Enfin, nous faisons le plan de ce mémoire.

I.1- Motivations

Les bases d'informations (bases de données, bases de connaissances, etc.) sont des systèmes utilisés pour représenter les croyances et connaissances que l'on peut avoir sur le monde réel ou tout au moins, sur une partie du monde réel. Cette représentation doit être la plus fidèle possible afin que la seule interrogation de ce système permette à son utilisateur de se faire une idée la plus exacte possible et de prendre des décisions adéquates.

La technologie des bases de données relationnelles, introduite par Codd **[COD 70],** est la plus répandue et se retrouve dans pratiquement tous les domaines d'application. Ce déploiement nécessite la présence d'informaticiens qualifiés pour administrer ces bases de données. Or, la complexité des systèmes de gestion de bases de données (SGBDs) augmente en même temps que leurs performances et leurs fonctionnalités. Ce qui fait que l'administrateur de bases de données (DBA) doit régler de plus en plus de paramètres pour une utilisation optimale du SGBD. La difficulté d'une telle tâche est largement reconnue alors que de nombreuses entreprises ne disposent pas d'un administrateur à plein temps. Dans ce contexte, la réduction des fonctions d'administration de bases de données est reconnue comme un nouveau challenge pour la communauté bases de données **[BER 98].**

L'administration *physique* des bases de données a été relativement étudiée comme par exemple dans le cadre du projet *AutoAdmin* de Microsoft pour la définition automatique d'index **[CHA 98]** ou la collecte de statistiques utiles pour l'optimiseur de requêtes

[GRA 98], **[CHA 00].** Ces approches sont basées sur l'étude de charges significatives de requêtes SQL accédant à un serveur de bases de données.

Pour l'administration *logique* (i.e. des schémas de relations) d'une base de données existante, cela nécessite de connaître et de comprendre la structure et la sémantique des données **[THA 95]**. Les informations sur la sémantique sont représentées entre autres par les *contraintes d'intégrité.* Ces dernières doivent être vérifiées à tout moment, i.e. pour toute instance de la base de données. Parmi les différents types de contraintes étudiées, les *dépendances fonctionnelles* (DFs) semblent être les plus importantes. Ces dernières jouent un rôle prépondérant lors du processus de *normalisation* **[LEV 99].** Dans le meilleur des cas, ces contraintes ont été spécifiées lors de la conception de la base de données et sont donc directement disponibles dans le SGBD. Malheureusement, ce cas idéal n'est pas toujours rencontré, soit parce que le SGBD ne supporte pas de telles spécifications, soit parce que le concepteur ne souhaite pas les préciser au moment où la base de données est créée. La découverte des dépendances fonctionnelles à partir d'une base existante est donc une piste pour analyser le schéma d'une base de données et simplifier les tâches d'administration « logique ».

D'un autre côté, en raison de la progression de techniques faciles et efficaces de collecte de données, le volume de données contenu dans les bases de données actuelles, augmente de manière astronomique (certaines relations peuvent contenir plusieurs centaines de milliers voire plusieurs millions de tuples). Mais notre capacité à en extraire des informations à forte valeur ajoutée reste à ce jour limitée, ce qui rend l'automatisation de cette tâche inévitable. Pour répondre à ces opportunités, le Data Mining est un domaine de recherche dédier à cette problématique. Il permet de développer des algorithmes performants utilisant des méthodes (ou techniques) dites intelligentes. Il existe à ce jour de nombreux travaux utilisant les techniques de Data Mining pour la découverte des dépendances fonctionnelles. Les plus importants travaux présents dans la littératures sont : TANE **[HUH 99]**, Dep-miner **[LOP 00]**, FastFDs **[WYS 01]** et FUN **[NOV 01]**.

I.2- Problématique

Cependant, les approches de découverte de dépendances fonctionnelles citées auparavant supposent que les données contenues dans les bases de données sont **précises**. Mais, en réalité ces données peuvent être imprécises, incomplètes, incertaines et/ou vagues, appelées de manière générique *données floues*. Par exemple, une donnée imprécise sur le salaire d'un employé « Ali » peut être représentée par "le salaire de Ali est d'environ 10000 *DA*". Un autre problème souvent rencontré dans les bases de données est celui des **valeurs manquantes** (en anglais : *missing values*). A titre d'exemple, dans le cas de gestion de prêts de bibliothèque, on a pu (parce qu'on a oublié ou parce qu'on ne le connaît pas) ne pas enregistrer le prénom de l'emprunteur d'un livre. Le modèle relationnel classique étant défini pour des données précises ne permet pas de gérer les données imprécises. Pour cela, plusieurs modèles sont apparus dans la littérature pour étendre le modèle relationnel aux données floues. Chacun de ces modèles traite un ou plusieurs aspects de la notion d'imprécision. Le plus important axe de recherche dans ce domaine est l'utilisation de la théorie des ensembles flous **[ZAD 65]** et la théorie des possibilités **[ZAD 78]** pour modéliser cette imprécision.

Les dépendances fonctionnelles, par leur utilité incontestable dans l'organisation logique des bases de données, ont aussi été étendues pour supporter les différentes situations d'imprécision des bases de données. Les définitions des dépendances fonctionnelles floues (DFFs) les plus rencontrées dans la littératures sont les définitions de Raju et Mazumder **[RAJ 86]**, Cubero et al **[CUB 93]**, Chen et al **[CHE 95]** et Bosc et al **[BOS 97b]**. Chacun des auteurs donne une sémantique bien précise à la notion de DFF selon sa vision de la *fuzzification*[1]. Certaines de ces définitions sont saines au sens qu'elles sont des règles implicatives et d'autres sont plus sémantiques au sens qu'elles véhiculent une sémantique similaire à celle des DFs. Mais, aucune définition ne regroupe les deux caractéristiques.

D'un autre côté, malgré la multitude de définitions des DFFs, nous avons constaté que le problème de la découverte des DFFs à partir des bases de données floues n'a pratiquement pas été abordé. A ce jour, il existe une seule approche de la découverte des

[1] Ce terme est employé ici pour désigner, non pas l'action de rendre une donnée floue mais plutôt, l'aspect flou de la donnée.

DFFs par des techniques de Data Mining, appelée approche de Wang **[WAN 00].** Cette approche est basée sur la généralisation de l'algorithme TANE, pour inférer des dépendances fonctionnelles floues en utilisant la définition des DFFs de Bosc **[BOS 97b],** mais elle reste tout de même non optimale pour des bases de données volumineuses.

Sur la base ces constatations, nous nous intéressons dans ce mémoire à la découverte des DFFs dans des modèles relationnels flous. Nos contributions à ce domaine sont exposées dans la section suivante.

I.3- Contributions

Dans ce mémoire, nous proposons une approche complète de découverte des dépendances fonctionnelles floues basée sur le concept de *stratification* des DFFs. La première contribution que nous avons formulé dans ce mémoire est une nouvelle définition des dépendances fonctionnelles floues, appelée dépendance fonctionnelle floue de niveau α, notée α-DFF. Cette définition regroupe les deux caractéristiques fondamentales des DFFs, présentes dans la littérature, elle est donc à la fois saine et sémantiques. Ensuite, nous avons prouvé un ensemble de règles d'inférence des α-DFFs, correspondant aux axiomes d'Armstrong. Le système d'Armstrong ainsi défini, nous servira à inférer seulement les dépendances fonctionnelles floues minimales non triviales valides dans la base de donnée considérée. Cela résulte du fait que les autres DFFs valides (obtenues par union, augmentation, décomposition ou transitivité) peuvent être retrouvées en utilisant ces règles d'inférence, similairement aux DFs classiques. La deuxième proposition est un processus complet d'Extraction de Connaissance à partir des Données dédié à la découverte des α-DFFs. Pour la première étape de ce processus consistant en la préparation des données cibles, nous avons proposé de faire un mapping des différentes représentation de données floues que nous pouvons retrouver dans les bases de données imprécises vers un format possibiliste (les valeurs imprécises sont modélisées par des *distributions de possibilité*). Ensuite, vient l'étape du Data Mining pour laquelle nous proposons une approche algorithmique appelée α-FFD-Extract. Cet algorithme se base sur la notion de coupe de niveau α pour l'élagage de l'espace des tuples afin de ne considérer que des tuples fortement ressemblant (ressemblance à un niveau supérieur ou égal à α, avec $\alpha \in]0, 1]$). Nous avons proposé deux lemmes. Le premier, nous permet de vérifier la validité des α-DFFs à tester quant au deuxième, il permet d'élaguer les coupes de niveau α

pour des ensembles d'attributs de taille supérieure à 1. L'élagage de l'ensemble des parties gauches candidates, pour une partie droite fixée d'une α-DFF, se fait en utilisant la notion de treillis, tel que les α-DFFs sont générées par ordre croissant de la taille des parties gauches [i,e. les DFFs de niveau *i* sont générées en utilisant le résultat du niveau (*i*-1)]. Par conséquent, le test de la minimalité des DFFs inférées est donc assuré. Les dépendances fonctionnelles floues ainsi inférées par notre approche sont minimales et non triviales. La validation des α-DFFs vérifiées dans une relation (base de données) revient à l'utilisateur (autrement dit, le DBA). Le paramètre α définit une espèce de *tuning* sur la ressemblance des tuples. Ce qui fait que le DBA à la liberté de régler ce paramètre selon le niveau des DFFs qu'il souhaite avoir.

I.4- Plan du mémoire

Le présent mémoire s'articule autour de cinq (05) chapitres. Dans ce chapitre introductif, nous avons présenté les motivations, la problématique et les principales contributions que nous apportons au domaine de la découverte des DFFs.

Le chapitre II, intitulé « les bases de données relationnelles » rappelle certaines notions de base du modèle relationnel classique présenté par D. CODD en 1970. Dans ce chapitre nous nous sommes basés sur les notions de décomposition, de formes normales et des dépendances fonctionnelles relatives aux formes normales. L'algèbre relationnelle n'étant pas l'objectif de ce mémoire, elle n'est pas présentée. Ce chapitre nous servira de support pour la présentation des différents modèles relationnels flous dans le chapitre IV.

Le chapitre III, intitulé « Extraction de Connaissances, Data Mining et Inférence des Dépendances Fonctionnelles », est divisé en quatre sections principales. Dans la première section, le domaine d'Extraction de Connaissances à partir des Données (ECD) est présenté tout en précisant le processus autour duquel il s'articule. L'étape centrale (Data Mining) de ce processus est ensuite définie en section 2, tout en citant les différentes tâches et méthodes du Data Mining. Parmi ces méthodes, ce trouve la méthode des règles d'association (RAs). Le modèle général des RAs et un algorithme d'induction de ces règles (l'algorithme Apriori) sont présentés dans la troisième section. Le choix de présenter la méthode des règles d'association n'est pas arbitraire, mais dirigé par le fait que les approches des règles d'association ont inspiré l'inférence des dépendances fonctionnelles

classiques. Ce dernier point fera l'objet de la dernière section de ce chapitre. Un état de l'art sur l'inférence (découverte) des dépendances fonctionnelles est alors présenté et deux des plus performantes approches en l'occurrence TANE et Dep-miner seront détaillées.

Le chapitre IV, intitulé « Théorie des ensembles flous, Bases de Données Floues et Dépendances Fonctionnelles Floues », traite en détail les bases de données floues. Dans ce chapitre, il y a cinq sections. La première et la deuxième rappellent les notions de bases de deux théories, la théorie des ensembles flous et la théorie des possibilités. La troisième section est une présentation non exhaustive des différents modèles relationnels flous existants dans la littérature. Dans la quatrième section, une sélection des plus importantes définitions de dépendances fonctionnelles floues est donnée. L'état de l'art sur l'inférence des dépendances fonctionnelles floues est donné en section quatre, dans lequel nous pouvons constater le manque flagrant de travaux sur ce sujet.

Les contributions que nous avons apporté au domaine de l'inférence des dépendances fonctionnelles floues se situent à plusieurs niveaux (définition d'une DFF, axiomatisation et processus de découverte de DFFs). Ces contributions sont présentées dans le chapitre V.

Enfin, nous terminons ce mémoire par une conclusion et nous présentons nos perspectives de recherches futures.

CHAPITRE 2

LES BASES DE DONNÉES RELATIONNELLES

Au jour d'aujourd'hui, le modèle relationnel **[COD 70]** est largement connu et diffusé à travers la communauté scientifique. Aussi, nous nous limiterons dans ce chapitre à donner uniquement les notions fondamentales relatives à notre contexte de travail.

II.1- Le modèle relationnel (principes et notations formelles)

Le modèle relationnel est fondé sur la théorie mathématique de relations. Cette théorie se construit à partir de la théorie des ensembles. Les notions les plus importantes pour introduire le modèle relationnel **[GAR 03]** sont : le domaine de valeurs, les relations et les attributs, définis comme suit.

Définition 1 : (Domaine)

Un domaine est un ensemble de valeurs caractérisé par un nom.

Les domaines sont donc les ensembles dans lesquels les données prennent leur valeur. On les note par Dom_i, (Exemple de domaine : Couleur_de_cheveux = {blanc, châtain, noir, roux}).

Définition 2 : (Relation)

Une relation est un sous-ensemble du produit Cartésien d'une liste de domaines caractérisé par un nom.

Une relation est donc composée de vecteurs. Une représentation plus commode d'une relation sous forme de table à deux dimensions a été retenue par CODD. On note la relation par r, tel que $r \subseteq Dom_1 \times \ldots \times Dom_n$. (où $\times$ dénote le produit Cartésien).

Définition 3 : (Attribut)

Colonne d'une relation caractérisée par un nom.

Le nom associé à un attribut est souvent porteur de sens. Il est donc en général différent de celui du domaine qui supporte l'attribut. (Exemples d'attributs : Nom, Prenom, Age, etc.). Les attributs sont dénotés par A_i, $i = 1..n$.

Définition 4 : (Schéma de relation)

Nom de la relation suivi de la liste des attributs et de la définition de leurs domaines.

Un schéma de relation est noté sous la forme : $\boldsymbol{R}(A_1 : \text{Dom}_1, A_2 : \text{Dom}_2, \ldots, A_n : \text{Dom}_n)$, ou plus simplement $\boldsymbol{R}(A_1, A_2, \ldots, A_n)$. Exemple : *Personne*(Nom, Prenom, Age).

Remarques :

- L'instance d'une relation appelée aussi extension, sera notée *r*. Par contre, le schéma d'une relation, noté ***R***, représente l'intension d'une relation,
- Nous noterons $t_j.A_i$ la restriction du tuple t_j à la valeur de l'attribut A_i.

II.2- Cohérence des données

La redondance d'information est indésirable dans une base de données dans la mesure où elle complique les mises à jour. Ainsi, lors de la modification d'une valeur répliquée, il est nécessaire de s'assurer que toutes les occurrences sont modifiées. Faute de quoi la base devient incohérente et à terme inutilisable. Les problèmes liés aux mises à jour (insertion, modification, suppression) sont appelés *anomalies de mise à jour*. Une démarche visant à éliminer (éviter) ces anomalies est appelée *décomposition sans perte d'informations*. Cette dernière est obtenue en définissant des *formes normales* sur la relation. Chaque forme normale correspond à un type particulier de *dépendances fonctionnelles*. Toutes ces notions seront présentées dans les sous-sections suivantes.

II.2.1- Décomposition relationnelle

L'approche par décomposition est utilisée lors de la conception de schémas relationnels. Le procédé de décomposition est comme suit : On part d'une relation composée de tous les attributs, appelée *relation universelle*, et on décompose cette relation en sous-relations ne souffrant pas des anomalies de mise à jour.

Définition 5 : (Relation universelle)

Table unique dont le schéma est composé de tous les attributs des tables constituant la base.

La définition de cette relation universelle suppose préalablement une nomination des attributs telle que deux attributs représentant la même notion aient le même nom et deux attributs représentant des notions distinctes aient des noms différents.

Le processus de décomposition est un processus de raffinement successif qui doit aboutir (du moins on l'espère) à isoler des entités et des associations élémentaires. Il doit être réalisé à partir d'une bonne compréhension des propriétés sémantiques des données. La compréhension de la théorie de décomposition des relations nécessite la bonne connaissance des deux opérations élémentaires de manipulation de relations que sont la *projection* et la *jointure*. En effet, nous allons décomposer par projection et recomposer par jointure.

Définition 6 : (Décomposition)

Une décomposition est un remplacement d'une relation $\boldsymbol{R}(A_1, \ldots, A_n)$ *par une collection de relations* $\boldsymbol{R}_1, \boldsymbol{R}_2, \ldots, \boldsymbol{R}_n$ *obtenues par des projections de* $\boldsymbol{R}$ *sur des sous-ensembles d'attributs dont l'union contient tous les attributs de* $\boldsymbol{R}$.

Définition 7 : (Décomposition sans perte)

Une décomposition sans perte d'information est une décomposition d'une relation $\boldsymbol{R}$ *en* $\boldsymbol{R_1}, \boldsymbol{R_2}, \ldots, \boldsymbol{R_n}$ *telle que pour toute extension* $\boldsymbol{r}$ *de* $\boldsymbol{R}$*, on ait* :

$$\boldsymbol{r} = \boldsymbol{r_1} \bowtie \boldsymbol{r_2} \bowtie \ldots \bowtie \boldsymbol{r_n} \qquad \textbf{(II.1)}$$

La décomposition sans perte d'information permet :

- d'assurer la non apparition de tuples excédentaires lorsqu'on effectue une jointure de relations dans la décomposition,
- de générer des résultats ayant du sens pour des requêtes requérant une jointure.

Le problème de la conception de bases de données relationnelles peut être perçu comme celui de décomposer la relation universelle composée de tous les attributs en sous-relations ne souffrant pas des anomalies vues ci-dessus, de sorte à obtenir une décomposition sans perte. Nous n'allons pas présenter les méthodes de décomposition, pour plus d'informations (voir **[ZAN 81][GAR 03]**).

Exemple 1 :

Soit la relation ***R***(Fournisseur, Rayon, Article) qui a pour instance la table suivante :

Fournisseur	**Rayon**	**Article**
Dupont	Ray1	Stylo
Durand	Ray1	Papier
Dupont	Ray2	Papier
Dupont	Ray1	Papier

Soient les trois relations $\boldsymbol{R_1}$, $\boldsymbol{R_2}$ et $\boldsymbol{R_3}$ définies par :

$\boldsymbol{R_1} = \prod_{\{\text{Fournisseur, Rayon}\}} (\boldsymbol{R})$,

$\boldsymbol{R_2} = \prod_{\{\text{ Rayon , Article }\}} (\boldsymbol{R})$,

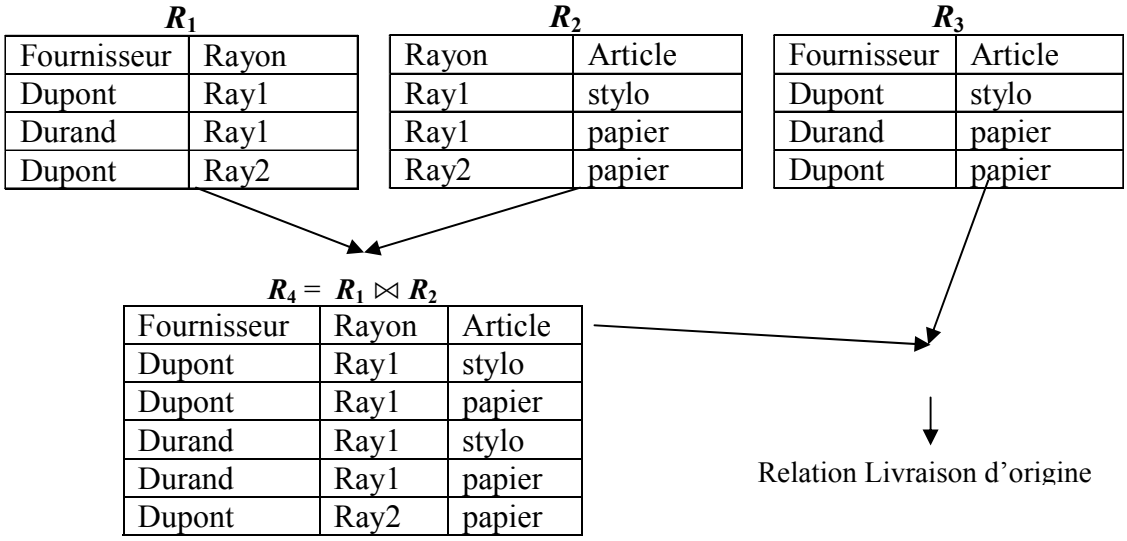
$\boldsymbol{R_3} = \prod_{\{\text{Fournisseur, Article}\}} (\boldsymbol{R})$.

où $\prod$ représente l'opérateur relationnel de projection.

Soient D_1 et D_2 deux décompositions de la relation ***R***, définies comme suit :

$D_1 = \{\boldsymbol{R_1}, \boldsymbol{R_2}\}$ et $D_2 = \{\boldsymbol{R_1}, \boldsymbol{R_2}, \boldsymbol{R_3}\}$.

La décomposition D_1 de ***R*** est une décomposition avec perte d'informations, puisque : l'extension de $\boldsymbol{R_4}$, $\boldsymbol{R_4} = \boldsymbol{R_1} \bowtie \boldsymbol{R_2}$ (où $\bowtie$ est l'opérateur relationnel de jointure naturelle) ne correspond pas à l'extension initiale de ***R***. Tandis que la décomposition D_2 est une décomposition sans perte d'informations, puisque : elle est composée de trois relations $\boldsymbol{R_1}$, $\boldsymbol{R_2}$ et $\boldsymbol{R_3}$, tel que la jointure de $\boldsymbol{R_1}$ et $\boldsymbol{R_2}$ donne $\boldsymbol{R_4}$, ensuite, la jointure de $\boldsymbol{R_4}$ et $\boldsymbol{R_3}$ donne l'extension de ***R*** définie au départ (voir schéma au dessous).



Afin d'exposer les différentes formes normales de relations, nous allons d'abord présenter les dépendances fonctionnelles.

II.2.2- Les dépendances fonctionnelles (DFs)

Les dépendances fonctionnelles représentent les contraintes d'intégrité les plus couramment rencontrées en bases de données. Ces contraintes sont essentielles dans la théorie relationnelle car elles sont à la base du calcul de clés et guident le processus de normalisation qui permet d'obtenir une organisation logique de la base. La notion de DF fut introduite avec le modèle relationnel par CODD afin de caractériser des relations pouvant être décomposées sans perte d'informations.

Définition 8 : (Dépendance fonctionnelle)

Soit $\boldsymbol{R}(A_1, A_2, \ldots, A_n)$ un schéma de relation, et X et Y des sous-ensembles de l'ensemble $\{A_1, A_2, \ldots, A_n\}$. *On dit que X détermine Y ou que Y dépend fonctionnellement de X, si et seulement si, pour toute extension r de* $\boldsymbol{R}$*, des valeurs identiques de X impliquent des valeurs identiques de Y*. On la note : $\boldsymbol{X \rightarrow Y}$.

Plus formellement,

$$\boldsymbol{X \rightarrow Y \Leftrightarrow \ (\forall\, t_1, t_2 \in r \ \ t_1.X = t_2.X \Rightarrow \ t_1.Y = t_2.Y)} \qquad \textbf{(II.2)}$$

II.2.2.1- Axiomes d'Armstrong

Les DFs obéissent à plusieurs règles d'inférences triviales. Les trois règles suivantes composent les axiomes des dépendances fonctionnelles et sont connues dans la littérature sous le nom d'axiomes d'Armstrong **[ARM 74]** :

Soient X, Y et Z trois ensembles d'attributs :

A_1 : (Réflexivité) : $Y \subseteq X \Rightarrow X \rightarrow Y$

Tout ensemble d'attributs détermine lui-même ou une partie de lui-même.

A_2 : (Augmentation) : $X \rightarrow Y \Rightarrow XZ \rightarrow YZ$ (tel que $XZ \equiv X \cup Z$)

Si X détermine Y, les deux ensembles d'attributs peuvent être enrichis par un même troisième.

A_3 : (Transitivité) : $X \to Y$ et $Y \to Z \Rightarrow X \to Z$

Cette règle est moins triviale et provient du fait que le composé de deux fonctions dont l'image de l'une est le domaine de l'autre est une fonction.

Théorème 1 :

A_1, **A_2**, **A_3** forment un ensemble correct et complet de règles d'inférences. □

A partir de ces trois règles, d'autres propriétés (règles) peuvent être déduites :

A_4 : (Union) : $X \to Y$ et $X \to Z \Rightarrow X \to YZ$

A_5 : (Pseudo transitivité) : $X \to Y$ et $WY \to Z \Rightarrow WX \to Z$

A_5 : (Décomposition) : $X \to Y$ et $Z \subseteq Y \Rightarrow X \to Z$

A partir de ces règles, il est possible d'introduire la notion de *dépendance fonctionnelle élémentaire* (DFE) **[ZAN 81]**.

Définition 9 : (Dépendance Fonctionnelle Elémentaire)

Une dépendance fonctionnelle élémentaire (DFE) est une dépendance fonctionnelle de la forme $X \to A$ qui remplit les deux conditions suivantes :

- A est un attribut élémentaire n'appartenant pas à X,
- Il n'existe pas $X' \subset X$ tel que $X' \to A$.

La seule règle d'inférence qui s'applique aux DFE est la transitivité.

II.2.2.2- Propriétés des DFs et définitions

A partir d'un ensemble de DFs, on peut composer par transitivité d'autres DFs. On aboutit ainsi à la notion de fermeture transitive d'un ensemble **F** de DFs. C'est l'ensemble des DFs considérées enrichi de toutes les DFs déduites par transitivité.

Soit **F** un ensemble de DFs sur un ensemble d'attributs ***R***. la fermeture de **F**, notée $\mathbf{F}^+$ est l'ensemble :

$$\mathbf{F}^+ = \{X \to Y : X, Y \subseteq \mathbf{R} \text{ et } \mathbf{F} \models X \to Y\} \qquad \textbf{(II.3)}$$

A partir de la notion de fermeture transitive, il est possible de définir l'équivalence de deux ensembles de DFs.

A) Equivalence d'ensembles de DFs

Soient **E**, **F** deux ensembles de DFs. E et F sont dits équivalents si et seulement si : $\mathbf{E}^+ = \mathbf{F}^+$.

Par suite, il est intéressant de déterminer un sous-ensemble minimal de DFs permettant de générer toutes les autres. C'est la *couverture minimale* d'un ensemble de DFs.

B) Couverture minimale

Une couverture minimale est un ensemble **F** de DFEs associé à un ensemble d'attributs vérifiant les propriétés suivantes :

- Aucune DFs dans **F** n'est redondante, ce qui signifie que pour toute DF f de **F,** **F -**$\{f\}$n'est pas équivalent à **F.**
- Toute DFE des attributs est dans la fermeture transitive de **F.**

Il a été montré que tout ensemble de DFs a une couverture minimale qui n'est pas (en général) unique.

C) Notion de clé

Soit $\boldsymbol{R}(A_1, A_2, ..., A_n)$ un schéma de relation, et X un sous-ensemble de $(A_1, A_2, ..., A_n)$, X est une clé de la relation $\boldsymbol{R}$ si et seulement si :

- $X \rightarrow (A_1, A_2, ..., A_n)$,
- X est minimal [il n'existe pas de $X' \subset X$ tel que $X' \rightarrow (A_1, A_2, ..., A_n)$].

Définition 10 : (Clé d'une relation)

Ensemble minimal d'attributs dont la connaissance des valeurs permet d'identifier un tuple unique de la relation considérée.

Remarque :

Un ensemble d'attributs qui inclut une clé, est appelé *superclé*.

II.2.3- Formes normales (1ère, 2ème et 3ème)

Les trois premières formes normales ont pour objectif de permettre la décomposition de la relation sans perte d'information, à partir de la notion de DF.

Définition 11 : (Première forme normale)

Une relation est en première forme normale (1ère FN) *si et seulement si tout attribut contient une valeur atomique*.

La première forme normale évite les domaines composés de plusieurs valeurs.

Définition 12 : (Deuxième forme normale)

Une relation est en deuxième forme (2ème FN) *normale si et seulement si :*

- *Elle est en première forme normale,*
- *Tout attribut n'appartenant pas à une clé ne peut pas dépendre d'une partie de cette clé*.

La deuxième forme normale élimine les anomalies créées par des DFs entre parties de clés et attributs non clé.

Définition 13 : (Troisième forme normale)

Une relation est en troisième forme normale (3ème FN) *si et seulement si* :

- *Elle est en deuxième forme normale,*
- *Tout attribut n'appartenant pas à une clé ne dépend pas d'un attribut non-clé.*

La troisième forme normale élimine les anomalies créées par des DFs entre les attributs non clé.

Afin d'éliminer les redondances crées par des DFs entre parties de clés et celles déjà éliminées par la 3ème FN, Boyce et Codd ont introduit la forme normale qui porte leur nom (en abrégé BCNF).

Définition 14 : (Forme normale de BOYCE-CODD)

Une relation est en BCNF si, et seulement si, les seules DFEs sont celles dans lesquelles une clé détermine un attribut.

Toutes les DFs sont de la forme $K \rightarrow A$, où K est une clé et A est un attribut non clé.

II.2.4- Quatrième forme normale et Dépendances multivaluée

La BCNF n'est pas suffisante pour éliminer complètement les redondances. Pour aller au-delà, il faut définir des dépendances plus lâches. Nous allons en voir de plusieurs types.

Définition 15 : (Dépendance MultiValuée)

Soit **R**(A_1, A_2, ..., A_n) une relation et X, Y deux sous-ensembles d'attributs de ***R***. *On dit que* $X \rightarrow\rightarrow Y$ *(X multivalue Y), si étant donné des valeurs de X, il y a un ensemble de valeurs de Y associées et cet ensemble est indépendant des autres attributs* Z = ***R*** - XY *de la relation* ***R***.

Plus formellement : $X \rightarrow\rightarrow Y$ est valide si et seulement si : pour chaque paire de tuples (t_1, t_2) de ***R***, il existe un tuple t_3 tel que :

$$(t_1.X = t_2.X = t_3.X \Rightarrow t_1.Y = t_3.Y \text{ et } t_2.Z = t_3.Z) \qquad \textbf{(II.4)}$$

avec Z = **R** - XY

II.2.4.1- Propriétés des DMVs

- Une DMV $X \rightarrow\rightarrow Y$ est triviale si $Y \subseteq X$ ou XY = ***R***,
- Les DFs sont des cas particuliers de DMVs (Si $X \rightarrow Y$ alors $X \rightarrow\rightarrow Y$),
- Si $X \rightarrow\rightarrow Y$ alors $X \rightarrow\rightarrow Z$, où Z = ***R*** - XY. Une notation plus intuitive serait : $X \rightarrow\rightarrow Y \mid Z$.

II.2.4.2- Règles d'inférence pour DF et DMV

Soient X, Y et $Z \subseteq$ ***R***.

I_0 : (Complémentation) : $X \rightarrow\rightarrow Y \Rightarrow (X \rightarrow\rightarrow (\boldsymbol{R} - \{X, Y\})$

I_1 : (Réflexivité) : $Y \subseteq X \Rightarrow X \rightarrow\rightarrow Y$

I_2 : (Augmentation) : $X \rightarrow\rightarrow Y \Rightarrow XZ \rightarrow\rightarrow YZ$

I_3 : (Transitivité) : $X \rightarrow\rightarrow Y$ et $Y \rightarrow\rightarrow Z \Rightarrow X \rightarrow\rightarrow (Z - Y)$

I_4 : (Conversion) : $X \rightarrow Y \Rightarrow X \rightarrow\rightarrow Y$

I_5 : (Interaction) : $X \rightarrow\rightarrow Y$ et $XY \rightarrow Z \Rightarrow X \rightarrow (Z - Y)$

Théorème 2 :

L'ensemble $\{A_1, A_2, A_3, I_0, I_1, I_2, I_3, I_4, I_5\}$ est correct et complet pour l'implication des DFs et DMVs considérées ensemble. □

Définition 16 : (Dépendance MultiValuée Elémentaire)

Une DMV $X \rightarrow\rightarrow Y$ d'une relation ***R*** est dite élémentaire si et seulement si :

- Y n'est pas vide et est disjoint de X,
- ***R*** ne contient pas autre DMV du type $X' \rightarrow\rightarrow Y'$ telle que $X' \subset X$ et $Y' \subset Y$".

Définition 17 : (Quatrième forme normale)

Une relation est en quatrième forme normale ($4^{ème}$ FN) *si et seulement si les seules dépendances multi-valuées élémentaire sont celles dans lesquelles une superclé détermine un attribut*.

Exemple 2 :

Soit la relation Enseignement(Cours, Enseignant, NotesCours), qui a pour instance la table suivante :

Cours	Enseignant	NotesCours
Physique	Dupont	Mécanique
Physique	Dupont	Thermodynamique
Physique	Martin	Mécanique
Physique	Martin	Thermodynamique
Mathématique	Durand	Algèbre
Mathématique	Durand	Géométrie

F = {Cours $\rightarrow\rightarrow$ Enseignant | NotesCours et {Enseignant, NotesCours} $\rightarrow$ Cours}.

Dans cet exemple chaque cours est associé à un ensemble d'enseignants et un ensemble de notes cours, et ces 2 ensembles (NotesCours et Enseignant) sont indépendants l'un de l'autre.

II.2.5- Dépendances de jointure (DJ) et Cinquième forme normale

Il y a des relations pour lesquelles une décomposition sans perte ne peut être réalisée qu'avec plus de deux (02) relations (cf. **Exemple 1**).

Définition 18 : (Dépendance de Jointure)

Soit $\boldsymbol{R}(A_1, A_2, ..., A_n)$ un schéma de relation et $\boldsymbol{R_1}$, $\boldsymbol{R_2}$, …, $\boldsymbol{R_m}$ des sous-ensembles de $\{A_1, A_2, ..., A_n\}$. *On dit qu'il existe une dépendance de jointure* (*DJ*)*, notée* $^*\{\boldsymbol{R_1}, \boldsymbol{R_2}, \ldots, \boldsymbol{R_m}\}$*, si* $\boldsymbol{R}$ *est la jointure de ses projections sur* $\boldsymbol{R_1}, \boldsymbol{R_2}, ..., \boldsymbol{R_m}$. Plus formellement :

$$^*\{\boldsymbol{R_1}, \boldsymbol{R_2}, \ldots, \boldsymbol{R_m}\} \Leftrightarrow \boldsymbol{R} = \prod_{R1}(\boldsymbol{R}) \bowtie \prod_{R2}(\boldsymbol{R}) \ldots \bowtie \prod_{Rm}(\boldsymbol{R}) \qquad \textbf{(II.5)}$$

Définition 19 : (Cinquième forme normale)

Une relation $\boldsymbol{R}$ *est en cinquième forme normale* ($5^{ème}$ FN) *si et seulement si toute DJ est triviale ou tout* $\boldsymbol{R_i}$ *est une superclé de* $\boldsymbol{R}$ (*c'est-à-dire que chaque* $\boldsymbol{R_i}$ *contient une clé de* $\boldsymbol{R}$).

Remarque :

- Une relation en 3FN dont toutes les clés sont simples est en 5NF.

II.3- Conclusion

Nous avons présenté dans ce chapitre les notions essentielles relatives à la conception de schémas de données et au maintien de la cohérence. La partie algèbre relationnelle relative à la manipulation de données (LMD) ne sera pas présentée dans ce mémoire vu qu'elle ne concerne pas notre travail.

Le modèle relationnel est sans doute un des modèles les plus utilisé par les éditeurs de SGBD (Oracle10, Informix, PostgreSQL, SQL Server, etc.). Il est basé sur une théorie solide qui permet une gestion efficace des bases de données parfaitement structurées. Mais en pratique, les bases de données que nous manipulons sont loin d'être parfaites, et souvent des imprécisions peuvent apparaître. De ce fait, la manipulation de cet aspect nécessite une reconsidération du modèle relationnel. Nous allons présenter ce type de base de données reconsidérée dans le chapitre IV.

Par contre, nous présentons dans le chapitre suivant l'utilisation du Data Mining pour la découverte de dépendances fonctionnelles.

CHAPITRE 3

EXTRACTION DE CONNAISSANCES, DATA MINING ET INFÉRENCE DES DÉPENDANCES FONCTIONNELLES

III.1- Extraction de Connaissances à partir des Données

L'**E**xtraction de **C**onnaissances à partir des **D**onnées (ECD), en anglais "**K**nowledge **D**iscovery in **D**atabases" (KDD), est une discipline dont l'objectif est la découverte automatique de connaissances nouvelles dans de très grands volumes de données **[FAY 96b]**. Dans la suite de ce mémoire, nous appellerons indifféremment ECD, KDD, Extraction de Connaissances à partir des Données, ou Knowledge Discovery in Databases pour désigner le même concept.

L'ECD est née des travaux croisés des chercheurs en statistiques, intelligence artificielle, reconnaissance de formes, base de données, apprentissage automatique, et autres domaines (cf. **Fig. III.1**).

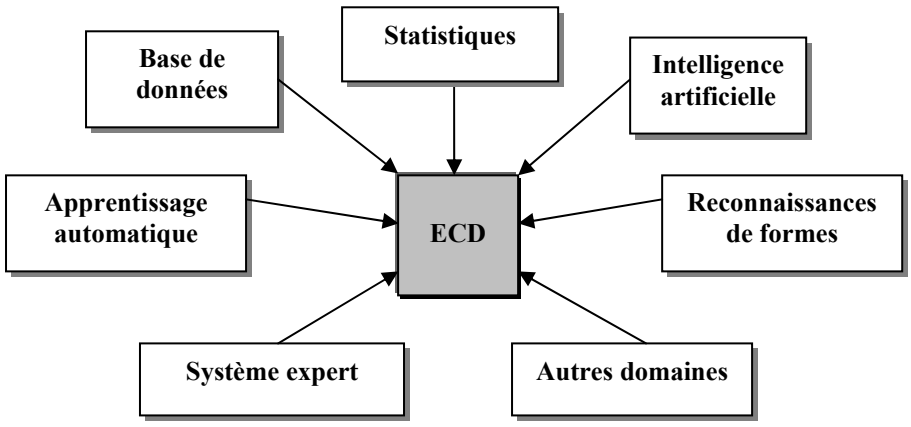


Fig. III.1. L'Extraction de Connaissances à partir des Données à la confluence de nombreux domaines **[BIA 01]**

III.1.1- Définitions de l'extraction de connaissances à partir des données

L'extraction de connaissances à partir des données a été définie par plusieurs auteurs, et tous s'accordent sur le fait qu'il s'agit d'un processus itératif et interactif de découverte de connaissances nouvelles et utiles (useful knowledge) à partir de données brutes (raw data) (cf. **Fig. III.2**). Nous donnons ci-après les définitions les plus usitées dans la communauté scientifique.

Fig. III.2. Transformation de données brutes en connaissances utiles

Définition 1 : (Knowledge Discovery in Databases)
Knowledge Discovery in Databases is the non trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data. **[FAY 96]**

Définition 2 : (Knowledge Discovery in Databases)
*The non trivial extraction of implicit, previously unknown, and potentially useful information from given data***. [FRA 92]**

Définition 3 : (Extraction de Connaissances à partir des Données)
L'Extraction de Connaissances à partir des Données est un processus itératif et interactif d'analyse d'un grand ensemble de données brutes afin d'en extraire des connaissances exploitables par un utilisateur-analyste qui y joue un rôle central. **[ZIG 01]**

III.1.2- Etapes du processus d'extraction de connaissances à partir des données (ECD)

Il existe dans la littérature plusieurs décompositions pour un processus d'ECD. Mais en réalité ces différentes décompositions représentent toutes un même processus **[MEG 04]**. Tout ce qui change d'un auteur à un autre est le degré de décomposition ou de

regroupement des étapes du processus. Dans **[FAY 91]** le processus d'ECD est représenté en quatre étapes qui sont : la consolidation, la sélection et le pré-traitement, la fouille de données et l'interprétation (cf. **Fig. III.3**). Nous présentons ci-après ces différentes étapes.

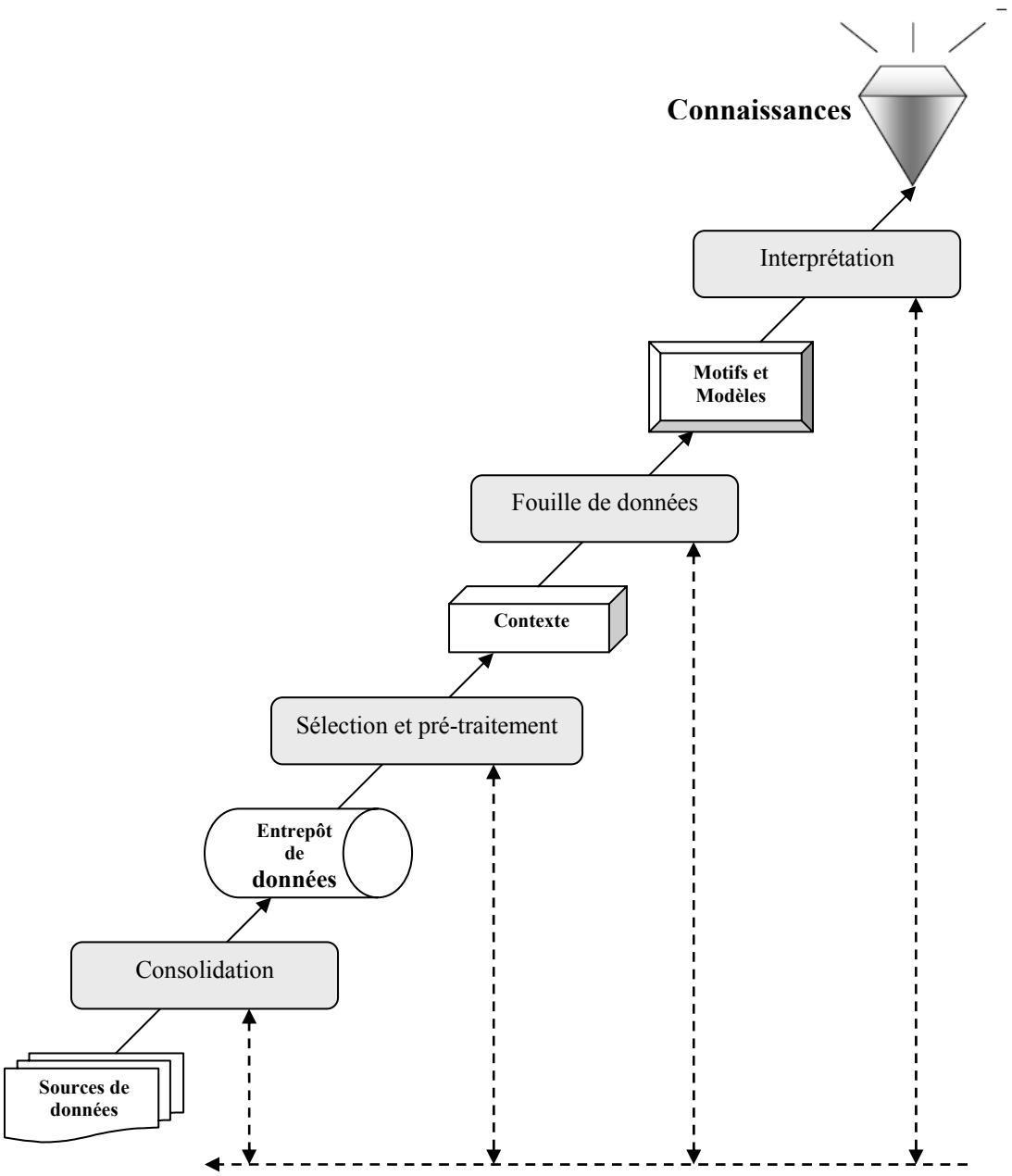


Fig. III.3. Etapes du processus d'Extraction de Connaissances dans les Données

III.1.2.1- Consolidation

Cette première étape consiste dans un premier temps en la collecte et le regroupement des données provenant de différentes sources (exemple : fichiers textes, différents SGBDs, ou bien même de notes manuscrites) au sein d'une seule et même base de données (ou entrepôt de données). Les données collectées sont souvent de nature diverses, autant du point de vue qualitatif que quantitatif (exemple : cours d'indices boursiers, données de recensement, logs web, séquences d'achats, etc.). Ces données peuvent être entachées d'erreurs (données aberrantes, imprécises ou manquantes) ce qui pousse dans un deuxième temps à les nettoyer, les mettre en forme et les coder selon un système uniforme pour pouvoir les exploiter dans les phases ultérieures.

Il existe plusieurs manières de procéder à la mise en forme des données. Nous nous intéressons à la génération de distributions de possibilité pouvant remplacer les valeurs de ces données. Notre proposition sur cet aspect particulier sera présentée dans le chapitre V.

III.1.2.2- Sélection et pré-traitement

Cette deuxième étape permet lors de la sélection, de choisir les données pertinentes à partir desquelles seront extraites les connaissances. Une fois les données sélectionnées, celles-ci sont pré-traitées. Il s'agit cette fois de représenter les données dans un format directement utilisable par les étapes suivantes du processus. On peut ainsi appliquer une opération de discrétisation sur les données (données discrètes). Par exemple, les cours d'indice boursiers peuvent être discrétisés en intervalles de valeurs. Dans le cas de variables discrètes, un travail de redéfinition des valeurs d'intervalles peut également être effectué (exemple : par union des intervalles de référence).

Cette seconde étape du processus d'ECD permet d'affiner les données pour permettre d'améliorer les résultats lors de l'interrogation dans la phase de fouille de données.

III.1.2.3- Fouille de données

Cette étape, également appelée « Data Mining », est considérée comme le cœur du processus d'ECD. En effet, elle permet le passage des données à l'expression des relations et des lois qui les sous-tendent. Cette étape est considérée comme la plus délicate du point de vu algorithmique. En effet, les ressources en temps et mémoire utilisées lors de cette

étape doivent rester raisonnables alors que les volumes de données traitées et les espaces de recherche sont très grands. L'utilisation active de contraintes (critères de sélection) lors de l'étape précédente permet de diminuer les espaces de recherche, et par conséquence les ressources en temps et en mémoire, et de cibler au mieux les traitements à effectuer.

III.1.2.4- Interprétation

Cette dernière étape est celle lors de laquelle l'utilisateur final intervient le plus. Il lui revient en effet la tâche d'analyser les résultats obtenus à l'issue de la fouille de données et de les interpréter en fonction du domaine de connaissances mis en jeu. Il faut cependant garder à l'esprit que cela n'enlève rien à l'importance que revêt l'intervention de l'utilisateur final lors des précédentes étapes. En effet, si les résultats sont jugés insuffisants lors de l'interprétation, l'utilisateur doit pouvoir remettre en cause les étapes qui selon lui ont été mal abordées, et à l'issue de chacune de ces étapes, il doit pouvoir porter le même regard critique.

Remarque :

- Le processus d' ECD est **itératif** car les résultats d'une étape peuvent remettre en cause les traitements effectués durant les étapes précédentes. Par conséquent, l'utilisateur peut décider de revenir en arrière à tout moment si les résultats ne lui conviennent pas. Ce processus est aussi **interactif** car la qualité des résultats obtenus dépend en grande partie de l'intervention des utilisateurs finaux, autrement dit les résultats obtenus sont liés aux différents choix que l'utilisateur est amené à effectuer.

III.2- Fouille de Données (Data Mining)

Par abus de langage, le terme Data Mining (fouille de données) est fréquemment confondu avec l'extraction de connaissances à partir des données (ECD). Mais en réalité comme on l'a vu précédemment, ce terme désigne dans son sens original seulement une étape du processus d'ECD. Cette étape consiste à **appliquer des méthodes dites intelligentes** sur les données afin d'en extraire des modèles (ou motifs).

Les méthodes de Data Mining peuvent être classifiées selon les tâches qu'elles accomplissent. Ses tâches sont détaillées dans la sous section suivante.

III.2.1- Les tâches du Data Mining

De nombreuses tâches peuvent être associées au Data Mining, parmi ces tâches nous pouvons citer **[PON 03] [TOM 00]** :

- La classification,
- L'estimation,
- La prédiction,
- Les associations,
- La segmentation,
- etc…

III.2.1.1- La classification

La classification consiste à créer une fonction qui classifie les données (ou individus) dans des classes prédéfinies selon des caractéristiques établies. Des exemples de tâches de classification sont :

- L'attribution ou non d'un prêt à un client,
- L'établissement d'un diagnostic médical,
- La classification d'image satellite,
- L'attribution d'un sujet principal à un article de presse,
- etc.

III.2.1.2- L'estimation

Elle consiste à estimer la valeur d'un champ à valeurs continues à partir des caractéristiques d'un objet. L'estimation peut être utilisée dans un but de classification. Il suffit d'attribuer une classe particulière pour un intervalle de valeurs du champ estimé. Des exemples de tâches d'estimation sont :

- La notation d'un candidat à un prêt. Cette estimation pourrait trouver une application dans l'attribution d'un prêt (classification), par exemple, en fixant un seuil d'attribution,
- L'estimation des revenus d'un client.

III.2.1.3- La prédiction

Cela consiste à estimer une valeur future. En général, les valeurs connues sont classées chronologiquement. On cherche à prédire la valeur future d'un champ. Cette tâche est proche des précédentes. Les méthodes de classification et d'estimation peuvent être utilisées en prédiction. Des exemples de tâches de prédiction sont :

- Prédire les valeurs futures d'actions,
- Prédire les cours d'indices boursiers,
- Prédire au vu de leurs actions passées les départs de clients.

III.2.1.4- Les associations

Cette tâche consiste à induire des corrélations entre les données. L'exemple type est la détermination des articles (le pain, le lait, les couches bébé, les jouets, les magazines pour enfants, etc.) qui se retrouvent ensemble dans le panier de la ménagère. Ce type d'application peut être utilisé à usage Marketing pour identifier des opportunités de vente croisée et concevoir des groupements attractifs de produits.

III.2.1.5- La segmentation

Il s'agit de créer des groupes homogènes dans la population (l'ensemble des enregistrements). Il appartient ensuite à un expert du domaine de déterminer l'intérêt et la signification des groupes ainsi constitués. Cette tâche est souvent effectuée avant les précédentes pour construire des groupes sur lesquels on applique des tâches de classification ou d'estimation.

Diverses techniques et méthodes peuvent être appliquées afin d'accomplir les tâches citées ci-dessus (classification, prédiction, segmentation, etc.). Ces techniques sont résumées dans la section qui suit.

III.2.2- Les méthodes et techniques de Data Mining

On peut mettre en évidence deux grandes catégories de méthodes dédiées à la fouille de données, à savoir les méthodes classiques et les méthodes intelligentes.

Parmi les méthodes classiques, on retrouve les outils déterministes de l'informatique ou des mathématiques tel que : les requêtes dans les bases de données (simples ou multi-critères), les statistiques descriptives, l'analyse de données : Analyse en Composantes Principales (ACP), etc.

Les méthodes intelligentes font appel à des techniques d'Intelligence Artificielle. Nous pouvons citer :

- La méthode des *k*-means (*k*-moyennes),
- La méthode des plus proches voisins,
- Les arbres de décision,
- Les réseaux de neurones,
- Les règles d'association,
- etc.

Les règles d'association étant la méthode utilisée dans le cadre de notre travail, la section suivante sera consacrée à présenter en détail cette méthode.

III.3- Les règles d'association

Les règles d'association (RAs) ont été initialement étudiées en analyse de données **[GUI 86],** puis étendues par **[AGR 93]** en fouille de données afin de trouver des régularités, des corrélations dans des bases de données de grande taille. Elles ont pour but de découvrir des relations *significatives* entre objets appartenant à une grande quantité de données. Ces règles ont d'abord été utilisées à usage marketing, mais elles sont actuellement utilisées dans tout autre domaine pour la recherche de cooccurrences fréquentes, si la structure des données s'y prête. Par exemple dans :

- La planification commerciale : elle consiste en l'identification des articles achetés fréquemment ensemble. Ce qui apporte une aide importante dans le placement des articles. Un problème proche de celui-ci est la définition de catalogues.
- Les réseaux de télécommunications : les bases de données d'alarmes détectées dans les réseaux de télécommunications sont constituées de rapports de situations anormales dans les composants des réseaux. Classiquement, ce sont plusieurs milliers d'alarmes

qui sont détectées chaque jour. Dans ce cadre, les règles d'association ont été utilisées avec succès dans le système TASA **[HAT 96][KLE 97]** pour le filtrage des alarmes non informatives, l'identification des causes d'anomalies et la détection et la prédiction d'anomalies.

- La recherche médicale : l'extraction de règles d'association dans les bases de données des patients permet d'apporter une aide sur le diagnostic, en identifiant les symptômes ou maladies précurseurs d'une maladie, une aide dans la définition de traitements en déterminant les symptômes ultérieurs ou les effets secondaires possibles, l'identification de populations à risque vis-à-vis de certaines maladies, etc.

- Le multi-média et internet : des quantités croissantes de données de diverses types (images, audio, vidéo, etc.), appelées données multi-média, sont stockées dans des bases de données dont le nombre ne cessent d'augmenter. L'extraction de règles d'association à partir de données multi-médias a donné lieu à de nombreuses études, principalement dans le cadre de l'analyse d'image. Les applications concernent la reconnaissance militaire, le filtrage des données parasites, la prévision météorologique, l'imagerie médicale, l'aide dans les enquêtes criminelles, etc. De même, un nombre important d'accès à ces données est réalisé chaque jour par des millions d'utilisateurs. L'extraction de règles d'association à partir des historiques des accès par les usagers aux ressources des sites Internet a été utilisée dans ce cadre pour l'aide à la conception et l'organisation des sites.

III.3.1- Modèle général des règles d'association

La découverte de règles d'association consiste à **induire** des relations entre les éléments d'un ensemble de données. Ce processus découvre des relations entre éléments de telle sorte que la présence de certains éléments tend à impliquer la présence d'autres éléments.

Formalisme

Afin de présenter le formalisme général des règles d'association **[AGR 94]** nous considérons les notations suivantes :

- *i*, un item (« un article » en français),

- $I = \{i_1, i_2, \ldots, i_n\}$, un ensemble de n items ou bien un *n*-itemset, l'ensemble I est supposé fini,
- t, une transaction (par exemple un ensemble d'achats). Une transaction est identifiée par un TID (Transaction IDentifier), et elle est constituée d'un sous ensemble I_0 d'items ($I_0 \subseteq I$),
- un sous ensemble $I_k \subseteq I$ de taille k est appelé un *k*-itemset,
- $T = \{t_1, t_2, \ldots, t_m\}$, un ensemble de m transactions d'items de I, l'ensemble T est supposé fini. Cet ensemble est appelé aussi *transaction database*,
- une transaction t_i contient un itemset I si et seulement si $I \subseteq t_i$.

Définition 4 : (Règle d'association)

Une règle d'association est une expression de la forme $A \Rightarrow C$. *où* $(A, C \subseteq I) \wedge (A, C \neq \varnothing) \wedge (A \cap C = \varnothing)$.

III.3.2- Sémantique d'une règle d'association

La règle $A \Rightarrow C$, signifie que "toute transaction de T contenant A tend à contenir C".

Les mesures habituelles pour évaluer les règles d'association sont le support et la confiance. Ces mesures sont calculées initialement pour un ensemble d'items (itemset) (i.e, un sous ensemble d'articles).

Définition 5 : (Le support)

"Soit I_0 un itemset, son support est le nombre de transactions t de la base T contenant I_0 par rapport au nombre total de transactions". Autrement dit, *c'est la probabilité qu'une transaction de T contienne* I_0.

$$Support\ (I_0) = \frac{|\{t \in T / I_0 \subseteq t\}|}{|T|} \qquad \textbf{(III.1)}$$

Le support d'une règle d'association $R : I_1 \Rightarrow I_2$ dans T est :

$$Support\ (I_1 \Rightarrow I_2) = Support\ (I_1 \cup I_2) = \frac{|\{t \in T / I_1 \cup I_2 \subseteq t\}|}{|T|} \qquad \textbf{(III.2)}$$

Définition 6 : (La confiance)

"*La confiance d'une règle d'association R* : $I_1 \Rightarrow I_2$ *est la probabilité conditionnelle qu'une transaction contienne* I_2 *sachant qu'elle contienne* I_1 ":

$$Confiance\ (I_1 \Rightarrow I_2) = \frac{Support\ (I_1 \cup I_2)}{Support\ (I_1)} \qquad \textbf{(III.3)}$$

On peut remarquer que :

$$Confiance\ (I_1 \Rightarrow I_2) = \frac{Support\ (I_1 \Rightarrow I_2)}{Support\ (I_1)} \qquad \textbf{(III.4)}$$

L'utilisation du signe $\Rightarrow$ est un abus de notation puisqu'il ne s'agit pas de l'implication logique classique (implication matérielle).

III.3.3- Inférence des règles d'association

Le problème de l'extraction de règles d'association consiste à déterminer l'ensemble des règles d'associations dont le support est au moins égal à un seuil minimum de support *MinSupp* et dont la confiance est au moins égale à un seuil minimum de confiance *MinConf*, définis par l'utilisateur.

Il existe actuellement trois grandes tendances pour induire les RAs dans une base de données, à savoir : l'approche orientée fréquents, l'approche orientée fermés et l'approche orientée maximaux. Du moment que l'approche à laquelle nous nous intéressons est l'approche orientée fréquents, cette dernière sera détaillée dans cette section.

Définition 7 : (Itemset fréquent, en anglais **Frequent ItemSet « FIS »)**

Soit T une base de transactions, I_0 un itemset et *MinSupp* un support minimal. Alors I_0 est considéré comme fréquent si seulement si : $Support\ (I_0) \geq MinSupp$.

Propriété d'anti-monotonie du support :

- Tout sous ensemble d'un ensemble fréquent est fréquent,
- Tout sur ensemble d'un ensemble non fréquent est non fréquent.

La découverte de règles d'association basée sur la recherche des itemsets fréquents FIS, est répartie en deux étapes :

- D'abord, trouver tout les itemsets ayant un support supérieur ou égal à *MinSupp* (les itemsets fréquents),
- Ensuite, à partir des FIS, engendrer l'ensemble des règles d'association ayant une confiance supérieure ou égale à *MinConf*.

Les plus importantes approches d'inférence des règles d'association utilisant les itemsets fréquents sont : *Apriori* **[AGR 94][SRI 96]**, *OCD* **[MAN 94]**, *AprioriTid* (amélioration d'Apriori), *Sampling* **[TOI 96b]** et *Partition* **[SAV 95]**. Nous avons choisi de détailler l'algorithme Apriori, pour le cas de la recherche des itemsets fréquents.

III.3.4- L'algorithme Apriori

L'algorithme Apriori a été proposé par Agrawal et Srikant **[AGR 94][SRI 96].** C'est un algorithme itératif pour la recherche des itemsets fréquents par niveaux, cela signifie que durant la $k^{ème}$ itération, un ensemble d'itemsets fréquents candidats de taille k est généré et un balayage du contexte est réalisé afin de supprimer les candidats infréquents. L'ensemble des k-itemsets fréquents ainsi générés est utilisé lors de l'itération k+1 suivante pour générer les candidats de taille k+1. Cet algorithme réalise donc m itérations afin de déterminer tous les itemsets fréquents dans l'ordre croissant de leur taille, m étant la taille des plus grands itemsets fréquents. Nous présentons dans la suite, l'algorithme Apriori ainsi qu'un exemple d'application de cet algorithme au contexte T.

Notations utilisées par Apriori

C_k	Ensemble de k-itemsets candidats (itemsets potentiellement fréquents). Chaque élément de cet ensemble possède deux champs : *itemsets* et *support*.
F_k	Ensemble des k-itemsets fréquents. Chaque éléments de cet ensemble possède deux champs : *itemsets* et *support*.

Durant la première itération de l'algorithme (ligne 1), tous les itemsets de taille 1 sont considérés et un balayage du contexte T est réalisé afin de déterminer l'ensemble F_1 des 1-itemsets fréquents. Chaque itération k suivante (ligne 2 à 16) se subdivise en deux phases. Durant la première phase (ligne 3), l'ensemble C_k des k-itemsets candidats est construit en effectuant la jointure de F_{k-1} sur lui-même ($C_k = F_{k-1} \bowtie F_{k-1}$). Cette phase est réalisée par la procédure Apriori-Gen. Durant la deuxième phase (ligne 4 à 16), un balayage du contexte est réalisé afin de déterminer le support de chacun des k-itemsets candidats présent dans C_k (ligne 7 à 13). Ensuite, les k-itemsets dont le support est supérieur ou égal à *MinSupp* (fréquents) sont insérés dans l'ensemble F_k. L'ensemble F de tous les itemsets fréquents reçoit l'union de tous les k-itemsets fréquents (ligne 17).

Algorithme 1 : Apriori

Entrée : T : une base de données transactionnelle, ε_α un seuil de fréquence absolu
Sortie : F : ensemble de tous les itemsets fréquents de T.

```
01 : k ←1 ; // déterminer l'ensemble F1 ; k ←2 ;
02 : Tant que Fk-1 ≠ ∅
     Faire
        // Générerl'ensemble Ck des candidats
03 :    Ck ← Apriori_Gen(Fk-1) ;
04 :    Pour tout e ∈ Ck
05         Faire
06 :          e.support ←0 ;
07 :       Pour tout transaction t ∈ T
           Faire
08 :          Pour chaque itemset s ∈ t de taille k
                 Faire
09 :                Si s ∈ Ck alors
10 :                   s.support + + ;
11 :                finsi
12 :             Fait ;
13 :       Fait ;
14 :       Fk ← {s ∈ Ck / s.support ≥ MinSupp} ;
15 :       k++ ;
16 :  Fait ;
17 :    Retourner F ← ∪k Fk
```

La génération de candidats dans Apriori se fait en utilisant un treillis. Une présentation détaillée des treillis est donnée en annexe 1. Nous donnons ci-après un exemple de treillis.

Exemple 1 :

Soit $I = \{A, B, C, D, E\}$ un ensemble d'items. Le treillis des itemsets potentiellement fréquents est représenté par le diagramme de Hasse représenté dans la figure **Fig. III.4**.

La découverte des itemsets fréquents consiste à extraire du *treillis des parties de l'ensemble I* des itemsets, dont le support dans le contexte d'extraction est supérieur ou égal au seuil minimal de support *MinSup*.

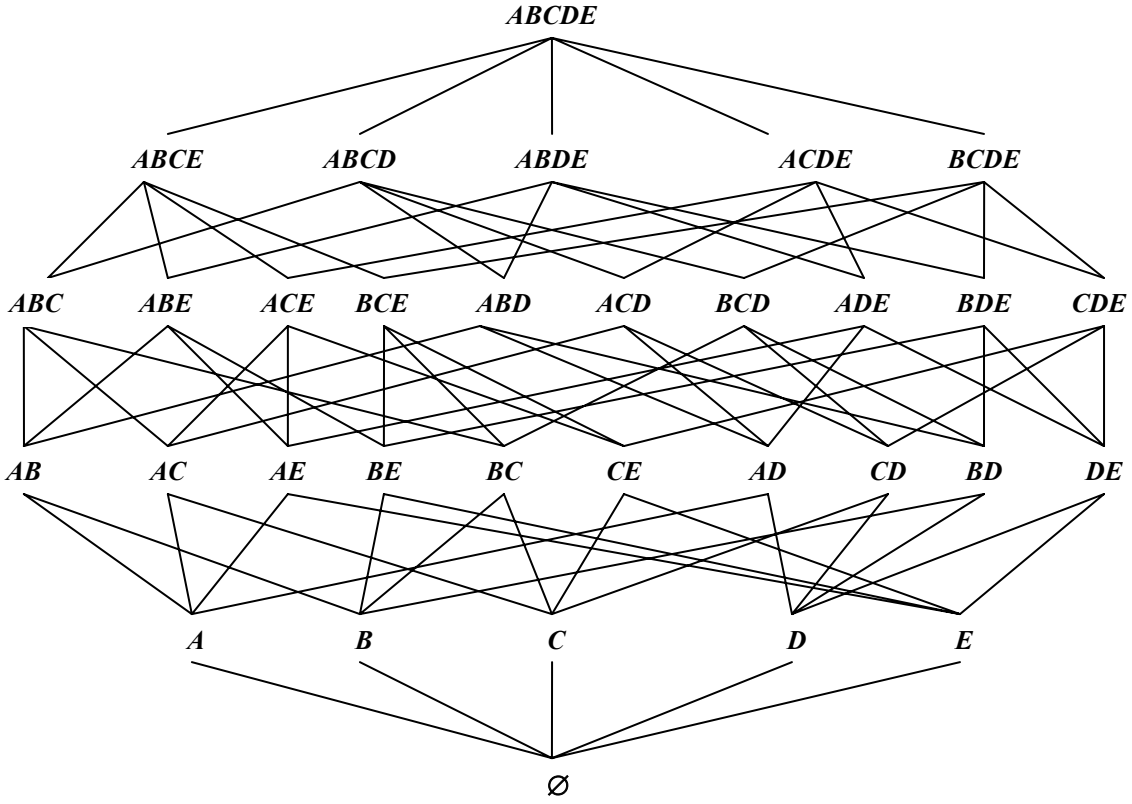


Fig. III.4. Trellis des fermés des itemsets potentiellement fréquents

Génération des candidats pour Apriori avec la procédure Apriori-Gen

La procédure Apriori-Gen reçoit un ensemble (F_{k-1}) de (k-1)-itemsets fréquents comme paramètre. Elle retourne un ensemble C_k de k-itemsets candidats qui est un sur-ensemble de l'ensemble des k-itemsets fréquents. Le pseudo-code de la procédure est présenté dans l'algorithme 2.

Durant la première partie de la procédure (ligne 1 à 5), deux (k-1)-itemsets fréquents p et q de F_{k-1} sont joints si les (k-2) premiers itemsets qui les composent sont identiques. Le résultat de cette jointure est un k-itemset potentiellement fréquent (candidat) qui est inséré dans C_k. Durant la deuxième partie (ligne 6 à 12), les k-itemsets candidats dont l'un des sous-ensembles s (de taille k-1) ne se trouvant pas dans F_{k-1} sont supprimés de C_k.

Algorithme 2 : Algorithme Apriori_Gen

Entrée : F_{k-1} : l'ensemble des (k-1)-itemsets fréquents
Sortie : C_k : l'ensemble des k-itemsets candidats

```
// Etape de jointure
1 : insert into Ck
2 : select p[1], p[2],…, p[k - 1], q[k]
3 : from Fk-1 p
4 : join Fk-1 q
5 : where p[1] = q[1], …, p[k-1] = q[k-1]
// Etape d'élagage
6 : for all itemset c ∈ Ck do
7 :     for all sub-set s de c where |s| = k-1 do
8 :         if s ∉ Fk-1 then
9 :             delete c from Ck
10 :        endif
11 :    endfor
12 : endfor
13 : return Ck
```

Exemple 2 : (application de l'algorithme Apriori sur un contexte *D*)

Soit un contexte d'extraction de règles d'association D constitué de six transactions, chacun identifié par son *TID*, et $I = \{A, B, C, D, E\}$, représenté dans la table suivante :

TID	*Items*			
1	A	C	D	
2	B	C	E	
3	A	B	C	E
4	B	E		
5	A	B	C	E
6	B	C	E	

L'application de l'algorithme Apriori au contexte *T* pour un seuil minimal de support *MinSup* de 2/6 est représentée dans la figure **Fig. III.5.** Quatre itérations sont exécutées par l'algorithme qui calcule quatre ensembles de candidats et d'itemsets fréquents et réalise quatre balayages du contexte.

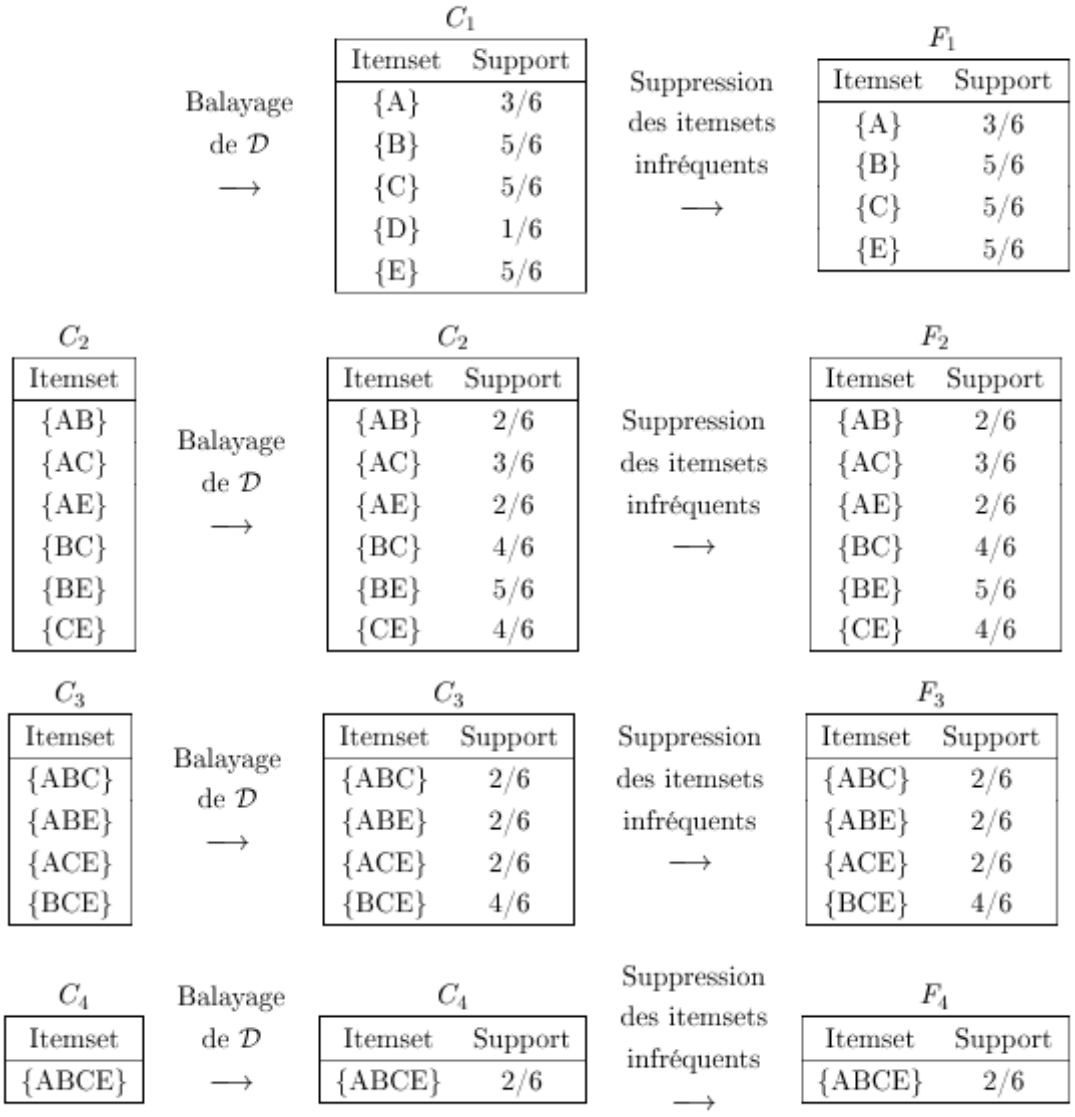


Fig. III.5. Extraction des itemsets fréquents dans le contexte *D* avec Apriori pour *MinSup* = 2/6

III.3.5- Critiques de la méthode des règles d'association

Les atouts :

- Méthode non supervisée à l'exception de la classification de différents articles en produits,

- **Clarté des résultats** : les règles sont faciles à interpréter,
- **Traite des données de taille variable** : le nombre de produits dans un achat n'est pas forcement défini,
- **Méthode adaptable** : on peut faire intervenir des produits virtuels (date, jour, saison) afin de « temporaliser » les données à analyser.

Les désavantages :

- **Pertinence des résultats** : ils peuvent être triviaux ou inutiles dans certains cas,
- **Efficacité faible dans certains cas** : pour les produits rares,
- **Traitement préalable des données** : classement des articles en produits,
- **Le coût calculatoire** de la méthode est important : Pour n transactions et k items donnés, la taille des tableaux de co-occurrence est : $n! / (n - k)! * k!$

Application des règles d'association :

Les règles d'association ont inspiré une nouvelle discipline à savoir, l'inférence de dépendances fonctionnelles à partir de bases de données existantes. La section suivante présente l'état de l'art sur cette nouvelle discipline.

III.4- Etat de l'art sur la découverte de DFs

Les approches de l'inférence des DFs et celles de l'inférence des RAs se ressemblent sur plusieurs points, et cela vient du fait que les deux règles :

- opèrent sur les mêmes objets de type (attribut, tuple),
- ont une forme semblable (Prémisse $\Rightarrow$ Conclusion).

Pour cela, certains auteurs se sont inspirés des méthodes de découverte des RAs (présentée dans la section III.3) pour induire des DFs.

Dans cette sous-section nous présentons d'abord l'intérêt d'inférer les DFs à partir de bases données existantes et nous détaillons ensuite deux des plus performantes approches d'extraction de DFs existantes dans la littérature. Ces approches sont inspirées bien sûr des techniques de Data Mining.

III.4.1- Intérêt de la découverte des dépendances fonctionnelles

Les dépendances fonctionnelles (cf. chapitre II) sont les contraintes d'intégrité les plus couramment rencontrées en base de données. Lors de la conception de systèmes d'information ou de bases de données, la connaissance de dépendances fonctionnelles est supposée acquise (après la phase d'analyse). Mais, durant la vie d'une base de données, le problème se pose dans d'autres termes : quelles sont, à partir des ensembles de données stockées, les dépendances effectivement vérifiées ? Ce problème est connu comme celui de l'inférence de dépendances fonctionnelles **[KIV 95][GOT 90],** et s'il a fait l'objet de nombreux travaux c'est parce que la réponse apportée à la question précédente est capitale dans plusieurs domaines d'application. Tout d'abord, dans la pratique, de nombreuses bases de données ne sont pas normalisées pour plusieurs raisons :

- originellement des erreurs ont pu être commises,
- des évolutions de schéma non maîtrisées au cours du temps ont pu avoir lieu,
- des choix d'organisation privilégiant l'optimisation des requêtes ont pu être fait.

Dans ces différents cas, l'administrateur de bases de données a besoin d'outils pour effectuer la ré-organisation logique des bases ou contrôler la satisfaction des dépendances fonctionnelles.

Ensuite, la re-ingénierie (ou *reverse engineering*) de bases de données s'appuie sur la connaissance des dépendances fonctionnelles valides. Il s'agit, à partir d'un schéma logique, de reconstituer un schéma conceptuel. Dans ce cadre, l'inférence de dépendances permet d'éliminer une contrainte forte, imposée par la plupart des approches existantes, selon laquelle le schéma de la base initial est bien conçu et normalisé **[PET 96].**

Il existe également plusieurs domaines d'application pour lesquels la connaissance des dépendances fonctionnelles valides est importante : ré-organisation physique de données, gestion de caches dans les applications clients/serveurs ou encore systèmes agents d'interrogation sur le web **[LOP 00][NOV 01].**

Enfin dans la gestion de datawarehouse (*entrepôts de données*) le problème est particulièrement à l'ordre du jour car lors de la modélisation des dimensions d'analyse

(hiérarchies de critères d'analyse), la représentation sous forme de relations fortement dénormalisées est préconisée.

III.4.2- Approches de découverte des DFs

Il existe plusieurs approches récentes traitant la découverte des dépendances fonctionnelles à partir de bases de données. Elles adoptent les principes de data mining pour proposer les algorithmes les plus efficaces permettant de répondre au problème posé. Les approches rencontrées dans la littérature sont les suivantes : TANE **[HUH 99],** DepMiner **[LOP 00],** FastFDs **[WYS 01],** FUN **[NOV 01]**.

Nous proposons d'abord d'exposer un algorithme naïf, ensuite, nous présentons les systèmes Dep-miner et TANE.

III.4.2.1- Algorithme naïf

L'algorithme 3 ci-après fonctionne de la manière suivante : A chaque attribut A de la relation, on lui associe tous les sous ensembles de $\mathbf{R} - \{A\}$ comme partie gauche. Le nombre de DFs testées est donc de l'ordre de $(|\mathbf{R}| * 2^{|\mathbf{R}|-1})$.

Algorithme 3 : Algorithme naïf **[MAN 94b]**

Entrée : r, une relation sur **R**
Sortie : F, une couverture de dep(r)

```
1 :  F ← ∅
2 :  for all A ∈ R do
3 :     for all X ⊆ R  do
4 :        if  r |= X→ A then
5 :           F ← F ∪ {X → A}
6 :        end if
7 :     end for
8 :  end for
```

Cet algorithme est loin d'être optimal pour différentes raisons, qui sont :

- la couverture obtenue est de très grande taille puisque nous obtenons toutes les DFs. Par exemple, aucune déduction n'est faite en utilisant les axiomes d'Armstrong à partir des DFs découvertes précédemment,
- l'algorithme est exponentiel par rapport au nombre d'attributs dans tous les cas.

III.4.2.2- Algorithme Dep-miner

L'algorithme Dep-miner **[LOP 00b]** détermine à la fois les DFs minimales non triviales et les relations d'Armstrong réelles. L'algorithme 4 présente le pseudo-code de Dep-miner (algorithme de découverte des DFs minimales non triviales valides dans une base de données).

Algorithme 4 : Dep-miner : algorithme général

Entrée : *r*, une relation.
Sortie : *F*, Dépendances fonctionnelles minimales et relations d'Armstrong réelles pour *r*.

1. **AGREE_SET** : calcul des ensembles en accord à partir de *r* ;
2. **CMAX_SET** : dérivation des compléments des ensembles maximaux à partir des ensembles en accord ;
3. **LEFT_HAND_SIDE** : calcul des parties gauches des DFs minimales à partir des compléments des ensembles en maximaux ;
4. **FD_OUTPUT** : afficher des DFs minimales
5. **ARMSTRONG_RELATION** : construction d'une relation d'Armstrong réelle à partir des ensembles maximaux.

Exemple 3 :

Soit la relation *Affectation*(id, empno, depno, annee, depnom, dir) sur laquelle nous allons dérouler les deux algorithmes de découverte de DFs (Dep-miner et TANE)

ID	*A*=empno	*B*=depno	*C*=annee	*D*=depnom	*E*=dir
1	1	1	85	Biochimie	5
2	1	5	94	Admission	12
3	2	2	92	Informatique	2
4	3	2	98	Informatique	2
5	4	3	98	Géophysique	2
6	5	1	75	Biochimie	5
7	6	5	88	Admission	12

Phase de prétraitement de l'algorithme Dep-miner

Cette phase est primordiale dans l'algorithme Dep-miner, elle permet de passer de l'espace des tuples c'est-à-dire toute la base de données à un espace plus réduit, celui de la base des partitions élaguées, qui correspond à l'ensemble des tuples ayant au moins une valeur commune avec un autre tuple pour un attribut donné. L'élagage dans cet algorithme est donc fait à l'avance par rapport aux autres algorithmes présents dans la littérature. L'algorithme 5 décrit les étapes de découverte de la base des partitions élaguées d'une relation *r*.

Algorithme 5 : Algorithme de prétraitement

Entrée : r, une relation sur R
Sortie : $\check{r}$, la base des partitions élaguées de la relation r

// Détermination de la classe d'équivalence du tuple $t_i \in r$ par rapport à l'attribut $A \in \mathbf{R}$, notée $[t_i]_A$, qui est définie par : $[t_i]_A = \{t_j \in r \ / \ t_i[A] = t_j[A]\}$

```
01:     for all  t_i ∈ r  do
02:         for all  A ∈ R  do
03:             [t_i]_A ← ∅
04:         endfor
05:     endfor
06:     for all  t_i ∈ r  do
07:         for all  A ∈ R  do
08:             if  t_i[A] = t_j[A]  then
09:                 [t_i]_A ← [t_i]_A ∪ {t_j}
10:             endif
11:         endfor
12:     endfor
```

// Soit π_A l'ensemble des classes d'équivalence qui représente une partition de r par rapport à l'attribut A. π_A est définie par : $\pi_A = \{[t]_A / t \in r\}$

```
13:     for all  A ∈ R  do
14:         π_A ← {[t]_A / t ∈ r}
15:     endfor
```

// Soit $\hat{}\pi_A$ la partition élaguée de la partition associée à l'attribut A est définie par : $\hat{}\pi_A = \{c \in \pi_A / |c| > 1\}$, où c est une classe d'équivalence.

```
16:     for all  A ∈ R  do
17:         ^π_A ← ∅
18:     endfor
19:     for all  A ∈ R  do
20:         for all  [t]_A ∈ A  do
21:             if  | [t]_A | > 1  then
22:                 ^π_A ← ^π_A ∪ [t]_A
23:             endif
24:         endfor
25:     endfor
```

// La base des partitions élaguées $\check{r}$ de la relation r correspond à l'union des différentes partitions élaguées des attributs de R. $\check{r}$ est définie par : $\check{r} = \cup_{A \in R} \hat{}\pi_A$

```
26:     ř ← ∅
27:     for all  A ∈ R  do
28:         ř ← ř ∪ ^π_A
29:     endfor
```

Remarque :

- Cette phase n'a pas été présentée ainsi dans l'algorithme Dep-miner, nous avons choisi de regrouper les différentes étapes sous forme d'un algorithme pour mieux les formalisées.

Application sur l'exemple :

- les partitions de chaque attribut de la relation sont :

π_A = {{1, 2}, {3}, {4}, {5}, {6}, {7}}, π_B = {{1, 6}, {2, 7}, {3, 4}, {5}},

π_C = {{1}, {2}, {3}, {4, 5}, {6}, {7}}, π_D = {{1, 6}, {2, 7}, {3, 4}, {5}},

π_E = {{1, 6}, {2, 7}, {3, 4, 5}}.

- les partitions élaguées sont :

$^\pi_A$ = {{1, 2}}, $^\pi_B$ = {{1, 6}, {2, 7}, {3, 4}},

$^\pi_C$ = {{1}, {2}, {3}, {4, 5}}, $^\pi_D$ = {{1, 6}, {2, 7}, {3, 4}},

$^\pi_E$ = {{1, 6}, {2, 7}, {3, 4, 5}}.

- la base des partitions élaguées associée à la relation *r,* est : $\check{r}$ = {$^\pi_A$, $^\pi_B$, $^\pi_C$, $^\pi_D$, $^\pi_E$}

Découverte des ensembles en accord de *r*

Les ensembles en accord de *r* sont découverts par l'algorithme 6, résultant du lemme 1 décrit ci-dessous. Cet algorithme opère de la façon suivante : en ligne 01 à 06, on calcule les classes d'équivalence maximales à partir de la base de partitions élaguées $\check{r}$. Puis, pour chaque classe d'équivalence maximale, tous les couples possibles sont générés (ligne 07 à 14). Les ensembles en accord sont alors calculés (lignes 15 à 23) pour tous les couples de tuples générés (car deux tuples de deux classes d'équivalence différentes sont en désaccord pour chaque attribut de ***R***). Finalement, l'ensemble des ensembles en accord de *r,* noté ag(*r*) est mis à jour (ligne 24 à 26).

Définition 8 : (Classe d'équivalence maximales)

Soit $\check{r}$ une base de partitions élaguées. L'ensemble *MC* des classes d'équivalence maximales de $\check{r}$ est défini comme suit : $MC = \text{Max}_{\subseteq} \{c \in ^\pi_A / \ ^\pi_A \in \check{r}\}$. Tel que *c* est une classe d'équivalence appartenant à la partition élaguée $^\pi_A$ de l'attribut *A*.

Lemme 1 : L'ensemble en accord d'une relation r correspond à l'union des ensembles en accord des couples de tuples appartenant à une même classe c de l'ensemble de classes d'équivalence maximales MC. Formellement : $\mathbf{ag}(r) = \cup_{c \in MC} \mathbf{ag}(c)$. □

Preuve

$c \in MC$ est une classe d'équivalence. Donc, c est un ensemble de tuples et ag(c) est bien défini.

(⊇) La classe d'équivalence c est un sous-ensemble de tuples de r. Donc, l'inclusion est évidente.

(⊆) Considérons un ensemble $X \in \text{ag}(r)$. Par définition, $\exists\ t_i, t_j \in r\ /\ \forall A \in X,\ t_i.A = t_j.A$. D'où, $\forall A \in X, t_i, t_j \in [t_j]_A \in {}^{\wedge}\pi_A$. Par définition de MC, $\exists\ c \in MC / t_i, t_j \in c$.

Donc $\text{ag}(r) \subseteq \cup_{c \in MC} \text{ag}(c)$. □

Algorithme 6 : AGREE_SET : calcul des ensembles en accord à partir de r.

Entrée : $\check{r}$, la base des partitions élaguées de la relation r
Sortie : ag(r), les ensembles en accord de r

```
01:  MC ← ř ;                    // Calcul des classes d'équivalence maximales MC de ř.
02:  for all  c ∈ ^π_A  do
03:        if  ∃ c' tel que  c ⊂ c'  then
04:              MC ← ř \ c ;
05:        endif ;
06:  endfor ;
07 :  ag(r) ← ∅ ;                // Calcul des ensembles en accord ag(r)
08 :  couples ← ∅ ;
09 :  for all classes d'équivalence maximales  c ∈ MC do
10 :      for all couples (t, t') ∈ c do
11 :          couples ← couples ∪ (t, t') ;
12 :          ag(r) ← ∅ ;
13 :       end for ;
14 :  end for ;
15 :  for all  ^π_A ∈ ř do
16 :       for all classes d'équivalence  c  ∈ ^π_A  do
17 :           for all (t, t') ∈ couples  do
18 :                 if  t ∈ c and  t' ∈ c  then
19 :                       ag(t, t') ← ag(t, t') ∪ {A};
20 :                 end if ;
21 :           end for ;
22 :       end for ;
23 :  end for ;
24 :  for all couples (t, t') ∈ couples do
25 :          ag(r) ← ag(r) ∪ ag(t, t') ;
26 :  end for ;
```

Application sur l'exemple :

- L'ensemble *MC* pour l'exemple précédent est : *MC* = {{1, 2}, {1, 6}, {2, 7}, {3, 4, 5}}
- à partir de *MC*, les couples générés sont : {{1, 2}, {1, 6}, {2, 7}, {3, 4}, {4, 5}, {3, 5}}

Le déroulement de l'algorithme est illustré par le tableau suivant dans lequel les colonnes montrent le traitement des couples pour les différentes classes d'équivalence. Notons que chaque colonne correspond à une itération du calcul des ensembles en accord (boucle entre 16 à 23).

Initialisation	$\{1, 2\} \in \pi_A$	$\{1, 6\} \in \pi_B$	$\{2, 7\} \in \pi_B$	$\{3, 4\} \in \pi_B$	$\{4, 5\} \in \pi_C$
$ag(1, 2) = \varnothing$	$ag(1, 2) = \boldsymbol{A}$	$ag(1, 2) = A$	$ag(1, 2) = A$	$ag(1, 2) = A$	$ag(1, 2) = A$
$ag(1, 6) = \varnothing$	$ag(1, 6) = \varnothing$	$ag(1, 6) = \boldsymbol{B}$	$ag(1, 6) = B$	$ag(1, 6) = B$	$ag(1, 6) = B$
$ag(2, 7) = \varnothing$	$ag(2, 7) = \varnothing$	$ag(2, 7) = \varnothing$	$ag(2, 7) = \boldsymbol{B}$	$ag(2, 7) = B$	$ag(2, 7) = B$
$ag(3, 4) = \varnothing$	$ag(3, 4) = \varnothing$	$ag(3, 4) = \varnothing$	$ag(3, 4) = \varnothing$	$ag(3, 4) = \boldsymbol{B}$	$ag(3, 4) = B$
$ag(3, 5) = \varnothing$	$ag(3, 5) = \varnothing$	$ag(3, 5) = \varnothing$	$ag(3, 5) = \varnothing$	$ag(3, 5) = \varnothing$	$ag(3, 5) = \varnothing$
$ag(4, 5) = \varnothing$	$ag(4, 5) = \varnothing$	$ag(4, 5) = \varnothing$	$ag(4, 5) = \varnothing$	$ag(4, 5) = \varnothing$	$ag(4, 5) = \boldsymbol{C}$

$\{1, 6\} \in \pi_D$	$\{2, 7\} \in \pi_D$	$\{3, 4\} \in \pi_D$	$\{1, 6\} \in \pi_E$	$\{2, 7\} \in \pi_E$	$\{3, 4, 5\} \in \pi_E$
$ag(1, 2) = A$	$ag(1, 2) = A$	$ag(1, 2) = A$	$ag(1, 2) = A$	$ag(1, 2) = A$	$ag(1, 2) = A$
$ag(1, 6) = B\boldsymbol{D}$	$ag(1, 6) = BD$	$ag(1, 6) = BD$	$ag(1,6) = BD\boldsymbol{E}$	$ag(1,6) = BDE$	$ag(1, 6) = BD$
$ag(2, 7) = B$	$ag(2, 7) = B\boldsymbol{D}$	$ag(2, 7) = BD$	$ag(2, 7) = BD$	$ag(2,7) = BD\boldsymbol{E}$	$ag(2, 7) = BD$
$ag(3, 4) = B$	$ag(3, 4) = B$	$ag(3, 4) = B\boldsymbol{D}$	$ag(3, 4) = BD$	$ag(3, 4) = BD$	$ag(3,4) = BD\boldsymbol{E}$
$ag(3, 5) = \varnothing$	$ag(3, 5) = \varnothing$	$ag(3, 5) = \varnothing$	$ag(3, 5) = \varnothing$	$ag(3, 5) = \varnothing$	$ag(3, 5) = \boldsymbol{E}$
$ag(4, 5) = C$	$ag(4, 5) = C$	$ag(4, 5) = C$	$ag(4, 5) = C$	$ag(4, 5) = C$	$ag(4, 5) = C\boldsymbol{E}$

- Les ensembles en accord suivants sont découverts :

$ag(1, 2) = A$, $ag(2, 7) = BDE$, $ag(3, 5) = E$, $ag(1, 6) = BDE$, $ag(3, 4) = BDE$, $ag(4, 5) = CE$.

D'où, nous obtenons $ag(r) = \{\varnothing, A, BDE, CE, E\}$.

Dérivation des compléments des ensembles maximaux

L'algorithme 7 permet le calcul des compléments des ensembles maximaux à partir des ensembles en accord. Le lemme 2 prouve sa correction.

Lemme 2 : $\text{Max}(\text{dep}(r)) = \text{Max}_{\subseteq} \{X \in \text{ag}(r) / A \notin X\}$. □

Tout d'abord nous calculons les ensembles maximaux pour chaque attribut de ***R*** (ligne 1 à 3) : pour un attribut A de R, les ensembles en accord qui ne contiennent pas A et qui sont maximaux par rapport à l'inclusion sont ajoutés aux ensembles maximaux (ligne 2). Trouver les compléments des ensembles maximaux cmax(dep(r), A) est trivial (ligne 4 à 9).

Algorithme 7 : CMAX_SET : dérivation des compléments des ensembles maximaux à partir des ensembles en accord

Entrée : les ensembles en accord sur r : ag(r)
Sortie : les compléments des ensembles maximaux : CMAX(dep(r))

```
01 :  for all A ∈ R do
02 :        max(dep(r), A) ← Max⊆ {X ∈ ag(r)/ A ∉ X}
03 :  endfor
04 :  for all attribute A ∈ R do
05 :        cmax(dep(r), A) ← ∅
06 :        for all X ∈ cmax(dep(r), A) do
07 :              cmax(dep(r), A) ← cmax(dep(r), A) ∪ (R / X)
08 :        endfor
09 :  endfor
```

Application sur l'exemple :

Appliqué à notre exemple, l'algorithme produit les résultats suivants :

max(dep(r), A)={BDE, CE}	cmax(dep(r), A)= {AC, ABD}
max(dep(r), B)={A, CE}	cmax(dep(r), A)= {$BCDE$, ABD}
max(dep(r), C)={A, BDE}	cmax(dep(r), A)= {$BCDE$, AC}
max(dep(r), D)={A, CE}	cmax(dep(r), A)= {$BCDE$, ABD}
max(dep(r), E)={A}	cmax(dep(r), A)= {$BCDE$}

Calcul des parties gauches des DFs

Pour trouver les parties gauches des DFs minimales à partir des ensembles maximaux, les notions d'hypergraphe et de transversal doivent être introduites (cf. voir **[MAN 94b]**).

Définition 9 : (Hypergraphe)

Une collection $\mathcal{H}$ de sous-ensembles de ***R*** est un *hypergraphe simple* si $\forall X \neq \emptyset$ et ($X, Y \in \mathcal{H}$ et $X \subseteq Y \Rightarrow X = Y$} **[BER 76]**.

Les éléments de $\mathcal{H}$ sont appelés les *arêtes* de l'hypergraphe et les éléments de ***R*** sont les *sommets* de l'hypergraphe. La collection cmax(dep(*r*), *A*) des compléments des ensembles maximaux max(*dep*(*r*), *A*) est un hypergraphe simple.

Définition 10 : (Transversal)

Un transversal T de $\mathcal{H}$ est un sous-ensemble de ***R*** dont l'intersection avec toutes les arêtes de $\mathcal{H}$ n'est pas vide, i,e. $T \cap E \neq \varnothing, \forall E \in \mathcal{H}$.

Un *transversal minimal* de $\mathcal{H}$ est un transversal T tel qu'il n'existe pas de transversal T', $T' \subset T$.

La collection des transversaux minimaux de $\mathcal{H}$ notée Tr($\mathcal{H}$). Les transversaux minimaux de $\mathcal{H}$ d'un hypergraphe simple sont reliés aux parties gauches des DFs minimales **[MAN 94b] [MAN 94c]** par :

$$Tr(cmax(dep(r), A)) = lhs(dep(r), A).$$

Notations utilisées par l'algorithme 8

$LHS_i[A]$	ensemble des parties gauches des DFs minimales non triviales de taille *i*.
lhs(dep(*r*), *A*)	ensemble des parties gauches des DFs minimales
L_{i+1}	ensemble des parties gauches candidates de taille (***i*+1**). $\mathbf{L_1} = \{A / A \in R\}$ $L_{i+1} = \{X / \lvert X \rvert = i + 1$ et $\forall\ Y \subset X$ et $\lvert Y \rvert = i$ on a $Y \in L_i\}$ et,

L'algorithme 8 initialise l'ensemble L_i avec les attributs présents dans cmax(dep(*r*), *A*) (ligne 3). La collection des transversaux minimaux est calculée comme suit (ligne 4 à 9) : pour chaque $l \in L_i$, nous testons si *l* est un transversal (ligne 5). Si c'est le cas, *l* est enregistré (ligne 5) dans LHS_i et est supprimé (ligne 6) de L_i (tous les sur-ensembles de *l* sont des transversaux non maximaux). Le niveau suivant est généré (ligne 7) en adaptant la fonction **Apriori-Gen.**

Algorithme 8 : LEFT_HAND_SIDE : calcul des parties gauches des dépendances fonctionnelles à partir des compléments des ensembles maximaux

Entrée : les compléments des ensembles maximaux : cmax(dep(r))
Sortie : les parties gauches des dépendances fonctionnelles minimales : lhs(dep(r))

1 : **for all** attribute $A \in R$ **do**
2 : $i \leftarrow 1$;
3 : $L_i \leftarrow \{B / B \in X, \ X \in \text{cmax}(\text{dep}(r), A)\}$;
4 : **while** $L_i \neq \emptyset$ **do**
5 : $LHS_i[A] \leftarrow \{l \in L_i / l \cap X \neq \emptyset, \forall X \in \text{cmax}(\text{dep}(r), A)\}$;
6 : $L_i \leftarrow L_i / LHS_i[A]$;
7 : $L_{i+1} \leftarrow \{l' / |l'| = i+1 \text{ et } \forall l / |l| = i, l \in L_i\}$;
8 : $i \leftarrow i + 1$;
9 : **end while;**
10 : $\text{lhs}(\text{dep}(r), A) \leftarrow \cup_i LHS_i[A]$;
11 : **end for ;**

Application sur l'exemple :

Pour l'attribut *A* :

Initialisation :	*Première itération*	*seconde itération*
$L_1 = \{A, B, C, D\}$	$LHS_1[A] = \{A\}$	$LHS_2[A] = \{BC, CD\}$
	$L_1 = \{B, C, D\}$	$L_2 = \{BD\}$
	$L_2 = \{BC, BD, CD\}$	$L_3 = \emptyset$

- Finalement, nous obtenons, les ensembles suivants :

lhs(dep(r), A) = {*A, BC, CD*}, lhs(dep(r), B) = {*AC, AE, B, D*}, lhs(dep(r), C) = {*AB, AD, AE, C*}, lhs(dep(r), D) = {*AC, AE, B, D*}, lhs(dep(r), A) = {*B, C, D, E*}

Inférence des DFs minimales non triviales

L'algorithme 9 affiche pour chaque attribut $A \in$ **R**, les DFs valides dans r et qui ont pour partie droite A, et cela en considérant les parties gauches de l'ensemble lhs(dep(r), A).

Algorithme 9 : FD_OUTPUT : Inférence des dépendances fonctionnelles minimales

Entrée : lhs(dep (r)) : les parties gauches
Sortie : F, ensemble de dépendances fonctionnelles minimales non triviales valide dans r.

1 : **for all** attributs $A \in R$ **do**
2 : **for all** $X \in \text{lhs}(\text{dep}(r), A) / X \neq \{A\}$ **do**
3 : Afficher $X \rightarrow A$
4 : **end for**
5 : **end for**

Application sur l'exemple :

Sur la relation de notre exemple, les DFs minimales non triviales suivantes sont vérifiées.

$r \models BC \rightarrow A$	$r \models AC \rightarrow C$	$r \models B \rightarrow D$
$r \models CD \rightarrow A$	$r \models AD \rightarrow C$	$r \models B \rightarrow E$
$r \models AC \rightarrow B$	$r \models AE \rightarrow C$	$r \models C \rightarrow E$
$r \models AE \rightarrow B$	$r \models AC \rightarrow D$	$r \models D \rightarrow E$
$r \models D \rightarrow B$	$r \models AE \rightarrow D$	

III.4.2.3- Algorithme TANE

L'algorithme TANE **[HUH 99]** découvre les dépendances fonctionnelles non triviales. L'approche est basée sur un algorithme par niveau qui commence en cherchant les dépendances fonctionnelles ayant une petite partie gauche (i.e. ayant le moins de chance d'être valides). Il partitionne l'ensemble des tuples d'une relation par rapport aux valeurs qu'ils prennent pour un ensemble d'attributs. Pour s'assurer qu'une DF est valide, il vérifie que, quand les tuples sont en accord sur sa partie gauche, ils sont aussi en accord sur sa partie droite.

Les partitions dans TANE

L'algorithme conserve des informations sur les tuples en accord sur un ensemble d'attributs. Pour cela, il utilise les notions de classe d'équivalence et de partitions présentées dans l'algorithme Dep-miner. Le lien entre partitions et dépendances fonctionnelles se fait par l'intermédiaire du concept de *raffinement*.

Définition 11 : (Raffinement)

Une partition π est un raffinement d'une partition π' si toute classe d'équivalence de π est un sous ensemble d'une classe d'équivalence de π'.

Le lemme 3 établit le lien entre DF et raffinement de partition.

Lemme 3 [COS 86][HUH 99] :

Une dépendance fonctionnelle $X \rightarrow A$ est valide si et seulement si π_X est un raffinement de π_A. □

Application sur l'exemple :

Dans notre exemple, pour tester la DF $D \rightarrow E$, nous considérons les partitions π_D = {{1, 6}, {2, 7}, {3, 4, 5}}et π_E = {{1, 6}, {2, 7}, {3, 4}, {5}}. On constate que chaque classe d'équivalence de π_D est incluse dans une classe d'équivalence de π_E d'où la DF $D \rightarrow E$ est valide.

Pour la DF $A \rightarrow B$, on a π_A = {{1, 2}, {3}, {4}, {5}, {6}, {7}}et π_B = {{1, 6}, {2, 7}, {3, 4}, {5}} : la classe d'équivalence {{1, 2}de π_A n'est incluse dans aucune des classes d'équivalence de π_B et donc $A \rightarrow B$ n'est pas vérifiée.

Le test peut encore être simplifié suivant le lemme 4.

Lemme 4 [HUH 99] : Une dépendance fonctionnelle $X \rightarrow A$ est valide si et seulement si $|\pi_X| = |\pi_{X \cup A}|$.□

Application sur l'exemple :

$|\pi_D| = 4 = |\pi_{DE}|$ d'où la DF $D \rightarrow E$ est valide. Et la DF $A \rightarrow B$ n'est pas vérifiée car $|\pi_A| = 6 \neq |\pi_{AB}| = 7$.

Pour améliorer les temps de traitement ainsi que l'occupation mémoire, deux optimisations sont utilisées : les partitions élaguées et la fonction g_3.

La fonction g_3 : cette fonction permet de mesurer l'erreur pour qu'une DF soit vérifiée : cette mesure représente le nombre de tuples à supprimer de la relation pour que la dépendance fonctionnelle devienne valide. On définit particulièrement $g_3(X)$ qui représente le nombre de tuples à supprimer de la relation pour que X devient une super-clé : $g_3(X) = 1 - |\pi_X| / |r|$. Cette dernière peut être calculée à partir des partitions élaguées par $g_3(X) = (||\hat{}\pi_X|| - |\pi_X|) / |r|$ où $||\hat{}\pi_X||$ est la somme des tailles des classes d'équivalence de $\hat{}\pi_X$. On peut remarquer que la valeur de la fonction est 0 lorsque l'ensemble sur lequel elle est appliquée est une super-clé.

Le lemme 4 peut être alors remplacer par le lemme suivant :

Lemme 5 : une DF $X \rightarrow A$ est valide si et seulement si $g_3(X) = g_3(X \cup A)$.□

Application sur l'exemple :

Pour tester le DF $D \rightarrow E$, on considère $\hat{}\pi_D$ = {{1, 6}, {2, 7}, {3, 4}}, $\hat{}\pi_D$ = {{1, 6},{2, 7},{3, 4}}. Comme $g_3(D) = (6-3)/7 = 3/7 = g_3(DE)$, la DF est vérifiée.

De même, la DF $A \rightarrow B$ n'est pas vérifiée car $g_3(A) = (2-1)/7 = 1/7 \neq g_3(B)=(0-0)/7 = 0$.

L'algorithme TANE est un algorithme par niveau : il commence par tester les singletons et parcourt le treillis des parties niveau par niveau. Pour un ensemble d'attributs X, il teste toutes les dépendances fonctionnelles de la forme $X - \{A\} \rightarrow A$ pour tout attribut $A \in X$. Une arête entre X et $X \cup A$ représente la dépendance fonctionnelle $X \rightarrow A$. de plus, l'élagage du treillis est effectué le plus tôt possible. L'algorithme 10 représente le pseudo-code de TANE.

Algorithme 10 : TANE

Entrée : r, une relation sur R
Sortie : F, une couverture de dep(r)

```
1 :   F ← ∅ ;
2 :   L0 ← {∅};
3 :   C+(∅) ← R ;
4 :   L1← {{A} | A∈ R};
5 :   i ← 1;
6 :   while Li ≠ ∅ do
7 :         COMPUTE-DEPENDENCIES (Li, F) ;
8 :         PRUNE (Li, F) ;
9 :         Li+1 ← GENERATE-NEXT-LEVEL (Li, F) ;
10 :        i ← i+1 ;
11 :  end while ;
```

La procédure COMPUTE-DEPEDENCIES

La procédure COMPUTE-DEPEDENCIES détermine les DFs $X - \{A\} \rightarrow A$ minimales, non triviale valides en se basant sur le lemme 6.

Définition 12 : (Ensemble de parties droites optimisées)

L'ensemble de parties droites optimisé $\mathbf{C^+(X)}$ est défini par :

$C^+(X) = \{A \in R \mid \forall B \in X, X - \{A, B\} \rightarrow B$ n'est pas valide$\}$.

Lemme 6 :

Soient $A \in X$ et $X - \{A\} \rightarrow A$ une DF valide. La DF $X - \{A\} \rightarrow A$ est minimale si et seulement si $\forall B \in X, A \in C^+(X/B)$. □

Remarque :

- Le lemme 6 montres que l'ensemble des parties droites optimisées $C^+(X)$ peut être utilisé pour tester la minimalité des DFs.

L'algorithme 11, présenté ci-dessous, représente le pseudo-code de la procédure COMPUTE-DEPENDENCIES.

Algorithme 11 : COMPUTE-DEPEDENCIES (L_i, F)

Entrée : L_i, les candidats de niveau i ; F, l'ensemble de dépendances fonctionnelles valides courant
Sortie : F, mise à jour avec les dépendances fonctionnelles du niveau L_i

```
1 :   for all X ∈ L_i do
2 :         C+(X) ← ∩ A∈X C+(X -{A});
3 :   end for;
4 :   for all X ∈ L_i do
5 :         for all A∈ X ∩ C+(X) do
6 :           if X −{A} → A est valide then
7 :                 F ← F ∪ {X-{A} → A};
8 :                 C+(X) ← C+(X) -{A};
9 :                 for all B ∈ R -{X} do
10 :                  C+(X) ← C+(X) -{B};
11:                 end for;
12:           end if;
13:         end for;
14:  end for;
```

La procédure PRUNE

La procédure PRUNE permet d'élaguer l'ensemble $C^+(X)$ en s'appuyant sur le lemme 7.

Lemme 7 : Soient $B \in X$ et $X - \{B\} \rightarrow B$ une DF valide.

- Si $X \rightarrow A$ est valide alors $X \cup B \rightarrow A$ est valide ;
- Si X est une super-clé alors $X - \{B\}$ est une super-clé.

La première partie de ce lemme permet de supprimer des candidats dans les ensembles de parties droites candidates (si $C^+(X) = \varnothing$ alors X est enlevé). On obtient donc $C^+(X)$, un ensemble de parties droites optimisé. La deuxième partie du lemme 4 permet d'élaguer tous les ensembles représentant une clé de la relation : en effet, une DF ayant une super-clé qui n'est pas une clé dans sa partie gauche ne peut pas être minimale.

Algorithme 12 : Prune (L_i, F)

Entrée : L_i, les candidates de niveau i ; F, l'ensemble de dépendances fonctionnelles valides courant
Sortie : L_i élaguer, F mise à jour avec les dépendances fonctionnelles du niveau L_i.

```
1 :   for all X ∈ L_i do
2 :     if C+(X) = ∅ then
3 :       L_i ← L_i - {X}
4 :     end if
5 :     if X est une super-clé then
6 :       for all A ∈ C+(X) - {X} do
7 :         if A ∈ ∩_{B∈X} C+(XA - {B}) then
8 :           F ← F ∪ {X→A}
9 :         end if
10 :      end for
11 :      L_i ← L_i - {X}
12 :    end if
13 :  end for
```

La procédure GENERATE-NEXT-LEVEL

La procédure GENERATE-NEXT-LEVEL détermine le niveau L_{i+1} à partir du niveau L_i. La procédure **PREFIX-BLOCKS(L_i)** partitionne L_i en des blocs disjoints comme suit : Soit un ensemble $X \in L_i$ contenant une liste d'attributs. Deux ensembles $X, Y \in L_i$ appartiennent tous deux au même bloc de préfixe si ils ont tous les deux un préfix commun de longueur (i-1).

Algorithme 13 : GENERATE-NEXT-LEVEL (L_i, F)

Entrée : PREFIX-BLOCKS(L_i), partition de L_i
Sortie : L_{i+1}, ensemble de candidates de niveau i+1

```
1 :   L_{i+1} ← ∅
2 :   for all K ∈ PREFIX-BLOCKS(L_i) do
3 :     for all {Y, Z} ⊆ K, Y≠ Z do
4 :       X ← Y ∪ Z
5 :       if for all A ∈ X, X/{A} ∈ L_i then
6 :         L_{i+1} ← L_{i+1} ∪ {X}
7 :   return L_{i+1}
```

Apports de TANE

- **Tester la validité :** pour tester la validité des dépendances fonctionnelles $X - \{A\} \rightarrow A$, l'algorithme a besoin des partitions élaguées $\hat{}\pi_{X / A}$ et $\hat{}\pi_X$. En fait, les partitions sont

calculées à partir de deux partitions déjà obtenues : par exemple, $\hat{}\pi_X$ est le produit de $\hat{}\pi_{X/A}$ et $\hat{}\pi_A$. Ce produit est la partition la moins raffinée qui raffine les deux partitions.

- **Tester la minimalité :** pour savoir si une DF $X\text{-}\{A\} \rightarrow A$ est minimale, il faut vérifier si la DF $Y\text{-}\{A\} \rightarrow A$ est valide pour un ensemble propre Y de X. cette information est conservée dans l'ensemble C(X -{A}) des parties droites candidates. Un attribut A se trouve dans C(X) s'il ne dépend d'aucun sous ensemble propre de X : c'est-à-dire que soit A est dans X et $X\text{-}\{A\} \rightarrow A$ n'est pas valide, soit A n'est pas dans X. pour tester la minimalité d'une DF, il suffit de tester la validité de la DF $X\text{-}\{A\} \rightarrow A$ où $A \in X$ et $A \in$ C(X -{B}), $\forall B \in X$.

- **Elagage :** l'algorithme par niveau permet de supprimer d'un coup un ensemble et tous ses sur-ensembles. Dans le cas présent, le fait que l'ensemble C(X) soit vide implique que C(Y) l'est aussi pour tout sur-ensemble Y de X. Donc, aucune DF de la forme $Y\text{-}\{A\} \rightarrow A$ ne peut être minimale.

III.5- Conclusion

Sachant que le domaine d'Extraction de Connaissances dans les Données est un domaine très vaste, ce chapitre n'a été en fait qu'un petit aperçu sur ce domaine. Par conséquent, les notions que nous avons voulu détailler sont celles qui s'apparentent à notre problématique, c'est-à-dire « l'inférence des dépendances fonctionnelles floues ».

Nous avons jusqu'ici présenté les principales approches existantes dans la littérature (Dep-miner et TANE) pour l'inférence des DFs. Les approches que nous avons ainsi présenté traitent de l'inférence de DFs pour des bases de données classiques.

Comme nous l'avons déjà fait remarquer, certaines bases de données peuvent contenir des informations imprécises, incomplètes et/ou manquantes. De telles bases de données sont appelées Bases de Données Floues (BDFs). De la même manière, les DFs résultantes de telles bases sont appelées Dépendances Fonctionnelles Floues (DFFs).

Dans le chapitre suivant, nous allons présenter les modèles de bases de données floues existants proposés par la communauté scientifique et au passage donner un résumé sur la théorie des ensembles flous et la théorie des possibilités. Enfin, nous présenterons les DFFs.

CHAPITRE 4

THÉORIE DES ENSEMBLES FLOUS, BASES DE DONNÉES FLOUES ET DÉPENDANCES FONCTIONNELLES FLOUES

Il existe différentes théories décrivant la modélisation et le raisonnement à partir de connaissances imparfaites. Les principales sont la théorie des probabilités, les fonctions de croyances **[SAF 76]**, la théorie des sous-ensembles flous **[ZAD 65]** et la théorie des possibilités **[ZAD 78]**. Les notions de base des deux dernières théories seront présentées dans les sections **§ IV.1** et **§ IV.2** respectivement. La deuxième partie de ce chapitre traite les principaux modèles flous de bases de données présents dans la littérature. Enfin, nous faisons un état de l'art exhaustif sur les différents modèles de dépendances fonctionnelles floues et les approches existantes pour la découverte de ces dernières.

IV.1- Théorie des ensembles flous

Du moment que nous avons retenu la théorie des ensembles flous et la théorie des possibilités comme cadre de modélisation de l'imprécision, nous présentons dans cette section de manière très synthétique les éléments de la théorie des ensembles flous nécessaires à la compréhension des modèles de bases de données floues. Afin de situer cette théorie dans le temps, nous donnons un petit historique sur l'émergence des ensembles flous dans différents domaines.

Dans la suite de ce chapitre, nous confondrons ensemble flou et sous-ensemble flou et utiliserons indifféremment ces deux appellations sachant qu'un sous-ensemble de manière générale est lui-même un ensemble.

IV.1.1- Historique sur la théorie des ensembles flous

L'histoire des sous-ensembles flous a connu de nombreux temps forts, dont certains des plus marquants sont donnés ci-après **[BOS 04]** :

- 1965 : publication du papier fondateur de Lotfi. A. Zadeh,

- Début des années 70 : suggestion d'une approche de type "système expert" en commande par des règles floues de la forme "**si** …**alors**…",
- 1973 : parution du premier livre sur les ensembles flous écrit par A. Kaufmann,
- 1975 : premiers résultats d'application des règles floues par Mamdani et Assilian,
- 1978 : introduction de la théorie des possibilités, nouveau cadre théorique de traitement de l'incertain fondé sur les ensembles flous,
- 1978 : parution du premier numéro de la revue internationale "Fuzzy Sets and Systems",
- Seconde moitié des années 70 : forte activité en commande floue en Europe (Angleterre, Danemark, Pays-Bas notamment) avec application à des processus industriels,
- 1985 : premier congrès de l'IFSA (International Fuzzy Systems Association) dont une édition a lieu depuis cette date tous les deux ans,
- Fin des années 80 : percée spectaculaire du flou au Japon avec de nombreuses applications industrielles allant du métro de Sendai aux photocopieurs en passant par les appareils électroménagers, les caméscopes, les appareils photo et les chaînes hifi,
- 1989 et 1990 : voyage d'étude français au Japon qui vont permettre la reconnaissance du domaine (club CRIN, groupe de travail OFTA) et conduire à la mise en place de la conférence francophone annuelle sur la logique floue et ses applications (LFA),
- Débuts des années 90 : développement important des applications en Allemagne avec soutien fort des pouvoirs publics,
- 1992 : première conférence internationale de l'IEEE (FUZZ-IEEE) à San Diego qui se déroule annuellement depuis,
- 1993 : parution du premier numéro de la revue internationale "IEEE Transactions on Fuzzy Systems",
- 1999 : création de la société savante européenne EUSFLAT.

IV.1.2- Eléments de base de la théorie des ensembles flous

La théorie des sous-ensembles flous a été introduite par Lofti Zadeh en 1965 **[ZAD 65]**. Zadeh étant automaticien, le domaine d'application initial des ensembles flous était la commande floue. Cependant, les ensembles flous ont reçu une attention dans de nombreuses autres communautés et spécialités, comme nous l'avons vu dans l'historique. Parmi les domaines utilisant cette théorie, se trouve la modélisation de la représentation humaine des connaissances. Ils sont utilisés soit pour modéliser l'imprécision, soit pour

représenter des informations sous forme linguistique assimilable par un système expert (les variables linguistiques).

Les sous-ensembles flous ont permis aussi d'améliorer les performances des systèmes de décision.

Remarques :

- Dans ce qui suit un ensemble classique est représenté par les lettres *A*, *B*, *C*, etc., et un sous-ensemble flou est représenté par les lettres *E, F*, *G*, etc,

Définition 1 : (sous-ensemble flou)

Soit *X* un univers de discours. Un sous-ensemble flou *E* dans *X* est entièrement défini par une fonction d'appartenance μ_F qui a chaque élément *x* de *X* associe un degré d'appartenance $\boldsymbol{\mu_E(x)}$ ($\boldsymbol{\mu_E(x)} \in [0, 1]$).

Un ensemble flou discret *E* peut être représenté comme suit :

$$\boldsymbol{E = \mu_E(x_1)/x_1+\mu_E(x_2)/x_2+\ldots+\mu_E(x_n)/x_n}$$

Ou bien $$\boldsymbol{E = \{\mu_E(x_1)/x_1, \mu_E(x_2)/x_2,\ldots, \mu_E(x_n)/x_n\}} \quad \textbf{(IV.1)}$$

Le degré d'appartenance $\mu_E(x)$ exprime la caractéristique de transition graduelle et non brutale entre l'appartenance complète et la non appartenance totale de l'élément *x* à l'ensemble *E*. Tel que :

$$\boldsymbol{\mu_E(x)} = \begin{cases} \mathbf{0} & \text{non appartenance totale } (x \text{ n'appartient pas du tout à } E). \\ \boldsymbol{t \in]0, 1[} & \text{plus } \mu_F(x) \text{ se rapproche de 1, plus } x \text{ appartient à } E. \\ \mathbf{1} & \text{appartenance complète } (x \text{ appartient complètement à } E). \end{cases}$$

Exemple 1 :

Supposons que nous voulions définir l'ensemble des personnes de «*taille moyenne*». En théorie des ensembles dite classique, nous conviendrons par exemple que les personnes de taille moyenne sont celles dont la taille est comprise entre 1m60 et 1m80. La fonction caractéristique de l'ensemble classique (cf. **Fig. IV.1.**) donne «0» pour les tailles hors de l'intervalle [1m60, 1m80] et «1» dans cet intervalle. C'est-à-dire qu'une personne mesurant

1m59 ne sera pas considérée de taille moyenne, alors qu'une personne plus grande d'un centimètre (1cm) sera considérée de taille moyenne.

Les ensembles flous ont été introduits pour exprimer une transition graduelle entre la non appartenance totale et l'appartenance complète.

Dans l'exemple 1, l'ensemble flou des personnes de *taille moyenne* sera ainsi défini par une « fonction d'appartenance » qui diffère d'une fonction caractéristique par le fait qu'elle peut prendre n'importe quelle valeur dans l'intervalle [0, 1]. A chaque taille possible correspondra un « degré d'appartenance » dans l'ensemble flou des *tailles moyennes* (cf. **Fig. IV.2**), compris entre 0 et 1.

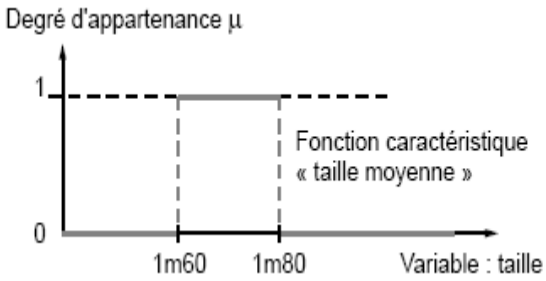


Fig. IV.1. Fonction caractéristique (Ensemble classique)

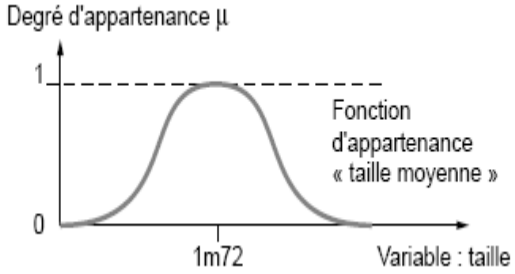


Fig. IV.2. Fonction d'appartenance (Ensemble flou)

Un ensemble flou comporte certains éléments caractéristiques, que nous présentons succinctement ci-dessous.

A)- Le noyau

Le noyau d'un sous-ensemble flou E de X, noté Noy(*E*), *est l'ensemble de tous les éléments qui lui appartiennent complètement*, il est représenté par :

$$\mathbf{Noy}(E) = \{x \in X \mid \mu_E(x) = 1\} \qquad \textbf{(IV.2)}$$

B)- Le support

Le support d'un sous-ensemble flou E de X, noté Supp(*E*), *est l'ensemble de tous les éléments qui lui appartiennent au moins un petit peu*, il est représenté par :

$$\mathbf{Supp}(E) = \{x \in S \mid \mu_E(x) > 0\} \qquad \textbf{(IV.3)}$$

C)- La hauteur

La hauteur d'un sous-ensemble flou E de X, notée h(*E*), *est la valeur maximale atteinte sur tout le support de E*, d'où :

$$\mathbf{h}(E) = \mathbf{Sup}_{x \in X}\ \mu_E(x) \qquad \textbf{(IV.4)}$$

La hauteur d'un ensemble flou *E* est définie aussi comme étant : "la borne supérieure de la fonction d'appartenance μ_E, aussi n'est-elle pas nécessairement atteinte".

Sous-ensemble flou normalisé

Un sous-ensemble flou est dit *normalisé* si sa hauteur est égale à 1.

D)- Coupe de niveau α (α-coupe)

La coupe (resp. coupe stricte) de niveau α de l'ensemble flou E noté E_α (resp. $E_{\bar{\alpha}}$) est l'ensemble usuel composé des éléments dont le degré d'appartenance à E est au moins égal (resp. strictement supérieur) à α. Elles sont définies comme suit :

$$E_\alpha = \{x / x \in X \text{ et } \mu_E(x) \geq \alpha\} \qquad \textbf{(IV.5)}$$

$$E_{\bar{\alpha}} = \{x / x \in X \text{ et } \mu_E(x) > \alpha\} \qquad \textbf{(IV.6)}$$

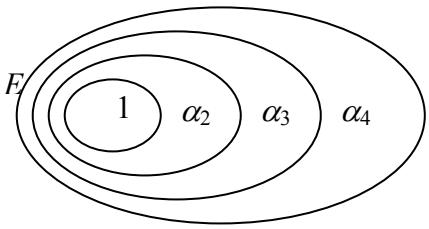
Les propriétés suivantes sont valides **[DUB 00]** :

- $E_1 = \text{noy}(E)$
- $E_{\bar{0}} = \text{Supp}(E)$

L'ensemble des coupes de niveau de *E* sont emboîtées au sens où :

- Si $\alpha > \alpha' \Rightarrow E_\alpha \subseteq E_{\alpha'}$

Une coupe de niveau α d'un ensemble flou a deux vues : une vue horizontale (cf. **Fig. IV.3**) et une vue verticale (cf. **Fig. IV.4**).



1, α_2 , α_3 , …, α_n sont les niveaux de coupe, tel que : **$1 > \alpha_2 > \alpha_3 > \ldots > \alpha_n$**

Fig. IV.3. Vue horizontale d'ensembles flous : α-Coupes

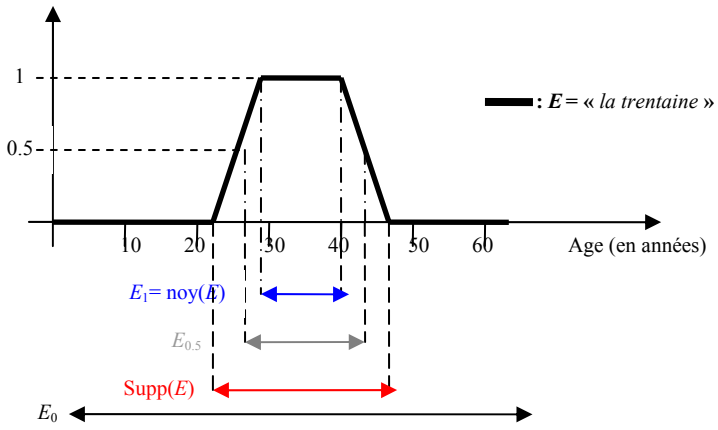


Fig. IV.4. Vue verticale d'une α-coupe sur l'ensemble flou normalisé « *la trentaine* »

Les coupes de niveau établissent un lien naturel entre ensemble flou et ensemble usuel qui permet d'exprimer un ensemble flou par ses interprétations plus ou moins exigeantes ou relaxées en termes de α-coupes. Une propriété intéressante des α-coupes est l'ensemble des éléments propres d'une coupe qui définissent une partition de l'ensemble flou.

Ensemble des éléments propres d'une coupe

L'ensemble des éléments propres d'une coupe, noté $\mathbf{P}_E(\alpha)$, est l'ensemble des éléments de E_α qui n'appartiennent à aucune coupe de niveau strictement supérieur à α, on a alors :

$$\mathbf{P}_E(\alpha) = \{x / x \in E_\alpha \text{ et } \forall \alpha' > \alpha,\ x \notin E_{\alpha'}\} \qquad \textbf{(IV.7)}$$

Un ensemble flou peut être reconstitué à partir des ses α-coupes en considérant les éléments propres de chaque coupe. L'ensemble initial est alors constitué des éléments de chaque ensemble $\mathbf{P}_E(\alpha)$ affectée du degré α.

IV.1.3- La cardinalité d'un ensemble flou

La cardinalité d'un sous-ensemble flou E de X, notée Card(E), est le nombre d'éléments appartenant à E pondéré par leur degré d'appartenance. Pour un ensemble flou E fini, on a donc :

$$\mathbf{Card}(E) = \sum_{x \in X} \mu_E(x) \qquad \textbf{(IV.8)}$$

De Lucas et Temini (1972) ont introduit cette définition et l'ont appelée « the power of fuzzy set ». Elle évalue « combien » d'éléments contient l'ensemble E. Cette définition suppose que :

- les degrés d'appartenance sont numériques,
- le support de E est fini.

Dans le cas de support infini, on peut utiliser l'intégrale de la fonction d'appartenance d'un ensemble flou sur son support, si il existe, on aura donc :

$$\mathbf{Card}\ (E) = \int_x \mu_E(x)\, dx \qquad \textbf{(IV.9)}$$

IV.1.4- Opérations ensemblistes floues

Les opérations usuelles définies sur les ensembles classiques (intersection, union, complémentation, etc.) ont été généralisées aux ensembles flous. Les opérations ensemblistes sur les ensembles flous sont généralement définies à partir des fonctions d'appartenance.

IV.1.4.1- Intersection et Union d'ensembles flous

Soient E, F deux ensembles flous. L'intersection (resp. l'union) des ensembles E et F est un ensemble flou G dont la fonction d'appartenance est définie à partir des fonctions d'appartenance de E et F comme suit :

$$\textbf{Intersection :}\ \forall x \in X,\ \mu_G(x) = \mu_{E \cap F}(x) = \mathrm{T}(\mu_E(x), \mu_F(x)) \qquad \textbf{(IV.10)}$$

$$\textbf{Union :}\ \forall x \in X,\ \mu_G(x) = \mu_{E \cup F}(x) = \perp(\mu_E(x), \mu_F(x)) \qquad \textbf{(IV.11)}$$

où : T désigne une T-norme

$\perp$ désigne une T-co-norme.

Normes et co-normes triangulaires

Les normes triangulaires (T-normes) et les co-normes triangulaires (T-conormes) permettent de généraliser les opérations ensemblistes d'intersection et d'union respectivement (et par extension la conjonctions et disjonction de propositions) sur les sous-ensembles flous. Pour que cette généralisation soit correcte, un certain nombre de propriétés doivent être vérifiées. Les définitions et les propriétés respectives des normes et co-normes sont données ci-après.

Définition 2 : (T-norme triangulaire)

La norme triangulaire est une fonction $T : [0, 1]\times[0, 1] \rightarrow [0, 1]$ qui pour tout élément $x, y, z, t \in [0, 1]$ possède les propriétés suivantes :

- Commutativité : $T(x, y)=T(y, x)$,
- Associativité : $T(x, T(y, z))=T(T(x, y), z)$,
- Monotonie : $T(x, y) \leq T(z, t)$ si $x \leq z$ et $y \leq t$,
- 1 est élément neutre : $T(x, 1) = x$,

De plus, elle assure que $\forall x, y \in [0, 1]$, $T(x, y) \leq x$ et $T(x, y) \leq y$,

Définition 3 : (T-conorme triangulaire)

La co-norme triangulaire est une fonction $\perp : [0, 1]\times[0, 1] \rightarrow [0, 1]$ qui pour tout élément $x, y, z, t \in [0, 1]$ possède les propriétés suivantes :

- Commutativité : $\perp(x, y) = \perp(y, x)$,
- Associativité : $\perp(x, \perp(y, z)) = \perp(\perp(x, y), z)$,
- Monotonie : $\perp(x, y) \leq \perp(z, t)$ si $x \leq z$ et $y \leq t$,
- 0 est élément neutre : $\perp(x, 0) = x$,

De plus, elle assure que $\forall x, y \in [0, 1]$, $\perp(x, y) \geq x$ et $\perp(x, y) \geq y$,

Il existe une infinité de normes et co-normes, les plus courantes sont données dans le tableau ci-dessous.

Norme: T(x, y)	Co-norme associée ⊥(x,y)	Nom
Min(x, y)	Max(x, y)	**Zadeh**
xy	$x+y-xy$	**probabiliste**
Max($x+y$-1, 0)	Min($x+y$, 1)	**Lukasiewicz**
xy / γ + (1-γ)($x+y-xy$)	$x+y-xy-xy$ (1- γ) / 1-(1-γ)xy	**Hamacher**
x si y =1 y si x =1 0 sinon	x si y = 0 y si x = 0 1 sinon	**Weber**

Tab. IV.1. Principales normes et co-normes triangulaires

Remarques :

- Le minimum est la plus grande T-norme de la famille des T-norme,
- Le couple Min/Max représente de nombreux avantages qui le rend particulièrement intéressant (exemple : la simplicité de calcul, distributivité de l'opération d'α-coupe par rapport à l'intersection et l'union, soit : $(E \cap F)_\alpha = E_\alpha \cap F_\alpha$ et $(E \cup F)_\alpha = E_\alpha \cup F_\alpha$,
- L'inconvénient du minimum (resp. maximum) est de ne pouvoir discriminer entre deux situations où des paires de degrés sont comparées avec une valeur minimale (resp. maximale) commune. Ainsi, Min(0.7, 0.2) = Min(0.3, 0.2) = 0.2

IV.1.4.2- Le complément

Le complément d'un ensemble flou F est défini habituellement par $F^c(x) = 1 - F(x)$ $\forall\, x \in X$. C'est la manière la plus simple pour exprimer l'évidente condition que les éléments dont le degré d'appartenance à F est le plus élevé, sont ceux dans le degré d'appartenance à son complément F^c est le plus bas. Le complément F^c du sous-ensemble flou F est défini par sa fonction d'appartenance de la manière suivante :

$$\forall\, x \in X,\ \mu_{F^c}(x) = 1 - \mu_F(x) \qquad \textbf{(IV.12)}$$

IV.1.4.3- Le produit Cartésien

Les problèmes considérés sont souvent décrits dans plusieurs univers de référence $X_1,\ldots,$ X_n. Il peut être intéressant de pouvoir raisonner dans un univers de référence X global composé de chacun des univers initiaux. X correspond donc au produit Cartésien de $X_1, X_2, \ldots,$ X_n :

$X = X_1 \times X_2 \ldots\ldots\ldots \times X_n$ et ses éléments x sont des n-uplets : $x = (x_1, \ldots, x_n)$.

Le produit Cartésien de n sous-ensembles flous $E_1,\ldots, E_n$, définis respectivement sur les univers de référence $X_1, \ldots, X_n$, est un sous-ensemble flou E définis sur X par sa fonction d'appartenance :

$$\forall\, x \in X, x = (x_1, \ldots, x_n) \in X, \boldsymbol{\mu_E(x) = \mathrm{Min}(\mu_{E1}(x_1), \ldots, \mu_{En}(x_n))} \qquad \textbf{(IV.13)}$$

IV.1.4.4- Les relations floues et la composition de relations

Soient deux ensembles de référence X et Y. On peut alors représenter une relation R entre X et Y par un sous-ensemble flou de $X \times Y$, dont la fonction d'appartenance μ_R est définie par : $\mu_R : X \times Y \rightarrow [0, 1]$. La relation R est notée $R(X, Y)$.

La composition de deux relations floues R_1 sur $X \times Y$ et R_2 sur $Y \times Z$ définit une relation $R = R_1 \circ R_2$ sur $X \times Z$ dont la forme la plus utilisée de la fonction d'appartenance μ_R est définie par :

$$\boldsymbol{\forall\, (x, z) \in X \times Z;\ \mu_R(x, z) = \mathrm{Sup}_{y \in Y}\, \mathrm{Min}(\mu_{R1}(x, y), \mu_{R2}(y, z))} \qquad \textbf{(IV.14)}$$

Définition 4 : (composition Sup-Inf)

Soient E un ensemble flou dans X. La composition sup-inf de relations floues $R(X, Y)$ est un ensemble flou F dans Y, notée par $F = R \circ E$. La composition sup-inf est définie par la fonction d'appartenance $\mu_{R \circ E}$ comme suit : $\mu_{R \circ E}(y) = \vee_X \{\mu_E(x) \wedge \mu_R(x, y)\}$.
où $\vee$ = Sup et $\wedge$ = Inf,

Soient $Q(X, Y)$ et $R(Y, Z)$ deux relations floues. La composition sup-inf $R \circ Q$ est une relation floue de X vers Z définie par la fonction d'appartenance $\mu_{R \circ Q}$ donnée par : $\mu_{R \circ Q}(x, z) = \vee_Y \{\mu_Q(x, y) \wedge \mu_R(y, z)\}$, $\forall\, (x, z) \in X \times Z$ et $\forall\, y \in Y$.

Définition 5 : (Relation inverse)

Soit R une relation floue sur $X \times Y$. son inverse R^{-1} est une relation floue sur $Y \times X$ tel que : $\forall\, (y, x) \in Y \times X$, $\mu_R(y, x) = \mu_R^{-1}(x, y)$.

Propriétés des relations floues

Une relation floue $R(X, Y)$ est dite :

1. **Réflexive** : Ssi $\forall\, x \in X$, $\mu_R(x, x) = 1$,
2. **Symétrique** : Ssi $\forall\, x_1, x_2 \in X$, $\mu_R(x_1, x_2) = \mu_R(x_2, x_1)$,
3. **T-Transitive** : $\forall\, x_1, x_2, x_3 \in X$, $\mu_R(x, z) \geq T(\mu_R(x, y), \mu_R(y, z))$.

Relations floues particulières

- Relation de ressemblance : Une relation floue est dite une relation de *ressemblance* (ou relation de *tolérance*) si elle est réflexive et symétrique. Deux exemples de relations de ressemblances que nous allons rencontrer dans la section **IV.4**, sont :

- $\mu_R(a, b) = \text{Min}_{x \in X}\ \ 1 - |\, \mu_a(x) - \mu_b(x)\,|$
- $\mu_R(a, b) = \text{Sup}_{x \in X}\ \ \text{Min}(\mu_a(x), \mu_b(x))$

- Relation de similarité : Une relation de similarité est une extension de la relation d'équivalence, enrichie de versions graduelles de la réflexivité, symétrie et transitivité (i,e. une relation de ressemblance qui vérifie la propriété de Min-transitivité **[DUB 00]**). La *Min-Trasitivité* est définie par : $\forall\, x_1, x_2, x_3 \in X$, $\mu_R(x, z) \geq \text{Min}(\mu_R(x, y), \mu_R(y, z))$

Remarques :

- Une relation de similarité peut être aussi définie avec Max-min-transitivité définie par :

Max-min-transitivité : $\forall\, x_1, x_2, x_3 \in X$, $\mu_R(x, z) \geq \text{Max}_{y \in X}\{\text{Min}(\mu_R(x, y), \mu_R(y, z))\}$

- La transitivité Max-min est une propriété très restrictive et elle ne s'applique pas sur tous les problèmes.

IV.1.4.5- L'égalité

Deux sous-ensembles flous E et F sont égaux si leurs fonctions d'appartenance sont égales en tout point de X :

$$\boldsymbol{E = F \Leftrightarrow \forall\, x \in X,\ \mu_E(x) = \mu_F(x)} \qquad \textbf{(IV.15)}$$

IV.1.4.6- L'inclusion

Une façon habituelle de définir l'inclusion d'un ensemble A dans un ensemble B repose sur l'expression : $(A \subseteq B) \Leftrightarrow (\forall x \in X, (x \in A) \Rightarrow (x \in B))$ **(IV.16)**
qui peut être écrit également en termes de contraintes entre valeurs des fonctions caractéristiques de A et B sous la forme : $(A \subseteq B) \Leftrightarrow (\forall x \in X, f_A(x) \leq f_B(x))$. Cette dernière s'étend canoniquement à deux ensembles flous E et F et conduit à :

$$(E \subseteq F) \Leftrightarrow (\forall x \in X, \mu_E(x) \leq \mu_F(x)) \quad \textbf{(IV.17)}$$

Une définition plus exigeante de l'inclusion est donnée par :

$$(E \subseteq F) \Leftrightarrow (\forall x \in X, (x \in \mathbf{Supp}(E)) \Rightarrow (x \in \mathbf{Noy}(F))) \quad \textbf{(IV.18)}$$

Cette définition présente un lien fort avec la notion de mesure de *nécessité* qui sera présentée dans la section théorie des possibilités.

IV.1.5- Les implications floues

D'ordinaire, l'implication $(P \Rightarrow Q)$ exprime une liaison entre les valeurs p et q des propositions P et Q. Elle rend *vrai* quand la proposition Q est vraie ou quand la proposition P est fausse, d'où les valeurs de vérité données dans la table **Tab. IV.2**.

P \ *Q*	**Vrai**	**Faux**
Vrai	Vrai	Faux
Faux	Vrai	Vrai

Tab. IV.2. Table de vérité de l'implication matérielle $P \Rightarrow Q$

Plusieurs approches d'extension de l'implication peuvent être envisagées lorsque les propositions P et Q prennent, non plus une valeur booléenne, mais une valeur de l'intervalle unité. Toute implication floue est une fonction (notée $\Rightarrow_f$) définie par :

$$\Rightarrow_f : [0, 1] \times [0, 1] \rightarrow [0, 1]$$
$$(p, q) \longmapsto (p \Rightarrow_f q) \quad \textbf{(IV.19)}$$

et elle d'autant plus vrai (resp. fausse) que son résultat est proche de 1 (resp. 0). Les implications floues peuvent être représentées selon plusieurs classifications. Dans la suite elles sont réparties en trois familles couvrantes les plus fréquentes (et utiles). Leurs principales caractéristiques sont précisées ainsi que leur sémantique afin d'en permettre un *usage rationnel*.

A)- S-implications

L'appellation S-implication vient de l'expression anglaise **S**trong **Implication**. On définit la classe des S-implications (notées $\Rightarrow_{s\text{-}i}$) à partir de l'expression : **((non *P*) ou *Q*)** de la manière suivante :

$$P \Rightarrow_{s\text{-}i} Q = \perp(1 - p, q) \qquad \textbf{(IV.20)}$$

où la disjonction (**ou**) est généralisée par une co-norme ($\perp$).

Il existe donc une infinité de S-implications et les trois plus communes sont données dans la table **Tab. IV.3**.

Nom	Symbole	Valeur de vérité	Co-norme sous-jacente
Kleene-Dienes	$\Rightarrow_{K\text{-}D}$	$\text{Max}(1 - p, q)$	$\perp(p, q) = \text{Max}(p, q)$
Reichenbach	$\Rightarrow_{Rb}$	$1 - p + p * q$	$\perp(p, q) = p + q - pq$
Lukasiewicz	$\Rightarrow_{Lu}$	$\text{Min}(1 - p + q, 1)$	$\perp(p, q) = \text{Min}(p + q, 1)$

Tab. IV.3. Les trois principales S-implications

De plus, l'ordre suivant sur les S-implications précédentes est valide.

$$(P \Rightarrow_{K\text{-}D} Q) \leq (P \Rightarrow_{Rb} Q) \leq (P \Rightarrow_{Lu} Q).$$

L'implication de Kleene-Dienes est la plus petite des S-implications (puisqu'elle est construite à partir de la plus petite des co-normes). La plus grande (notée $\Rightarrow_{S\text{-}M}$) est trouvée en prenant la co-norme de Weber, soit :

$$P \Rightarrow_{S\text{-}M} Q = \begin{cases} \mathbf{(1 - p)} & \text{si } q = 0 \\ \mathbf{q} & \text{si } p = 1 \\ \mathbf{1} & \text{sinon} \end{cases}$$

Remarques :

- Comme pour l'implication usuelle, on a : $(P \Rightarrow_{\text{s-i}} \textbf{faux}) = \perp(1 - p, 0) = 1 - p$.
- Les implications de cette classe sont leur propre contraposée car d'après la formule **(IV.19)**, on a : **non** $Q \Rightarrow_{\text{s-i}}$ **non** $P = \perp(1 - (1 - q), 1 - p) = \perp(q, 1 - p) = \perp(1 - p, q) = (P \Rightarrow_{\text{s-i}} Q)$

B)- R-implication

La seconde classe d'implications floues regroupe les R-implications, ainsi dénommées parce qu'elles utilisent le principe de *résiduation*[2].

Dans le cas classique on a : (P **et** ($P \Rightarrow Q$)) $\Leftrightarrow$ (P **et** (**non** P **ou** Q)) $\Leftrightarrow$ ((P **et non** P) **ou** (P **et** Q)) $\Leftrightarrow$ (P **et** Q). Ce qui fait que la valeur de vérité de la proposition (P **et** ($P \Rightarrow Q$)) est inférieure ou égale à celle de la proposition Q, i,e., $(p$ **et** $(p \Rightarrow q)) \leq q$).

Dans le cas flou cette inégalité sert de point de départ pour chercher l'élément maximal la vérifiant, soit :

$$\boldsymbol{P \Rightarrow_{\text{R-i}} Q = \text{Sup}_{[0,\,1]} \{u \mid \text{T}(p, u) \leq q\}} \qquad \textbf{(IV.21)}$$

T étant une norme triangulaire.

Il y a donc une infinité de R-implications dont la forme générale est :

$$(P \Rightarrow_{\text{R-i}} Q) = \begin{cases} 1 & \text{si} \quad p \leq q \\ \text{T}(p, q) & \text{sinon} \end{cases}$$

Les plus courantes figurent dans la table **Tab. IV.4** avec la norme triangulaire (génératrice) qui leur est associée. L'implication de Rescher-Gaines peut être vue comme une R-implication particulière prenant en argument des propositions floues en rendant un résultat booléen, dérivant d'une formule voisine de la formule **(IV.21)**, à savoir :

$$P \Rightarrow_{\text{RG}} Q = \text{Sup}_{\{0,\,1\}} \{u \mid \text{T}(p, u) \leq q\}$$
$$= \begin{cases} 1 & \text{si } p \leq q \\ 0 & \text{sinon.} \end{cases}$$

[2] Traduit du terme anglais : "*Residual implication*"

et ce, pour toute norme T. De façon analogue aux S-implications, ces implications peuvent être ordonnées : $(p \Rightarrow_{\text{R-G}} q) \leq (p \Rightarrow_{\text{Gö}} q) \leq (p \Rightarrow_{\text{Gg}} q) \leq (p \Rightarrow_{\text{Lu}} q)$.

Nom	Symbole	Valeur de vérité	Norme sous-jacentes
Gödel	$\Rightarrow_{\text{Gö}}$	$\begin{cases} 1 & \text{si } p \leq q \\ p & \text{sinon} \end{cases}$	$T(p, q) = \min(p, q)$
Goguen	$\Rightarrow_{\text{Gg}}$	$\begin{cases} 1 & \text{si } p \leq q \\ q / p & \text{sinon} \end{cases}$	$T(p, q) = p\ q$
Lukasiewicz	$\Rightarrow_{\text{Lu}}$	$\begin{cases} 1 & \text{si } p \leq q \\ 1 - p + q & \text{sinon} \end{cases}$	$T(p, q) = \max(p+ q - 1, 0)$

Tab. IV.4. les trois principales R-implications

L'implication de Gödel est la plus petite R-implication propre (i,e., en excluant celle de Rescher-Gaines), puisque elle est construite à partir de la plus grande norme triangulaire. La plus grande (notée $\Rightarrow_{\text{R-M}}$) est celle construite à partir de la plus petite norme (celle de Weber), soit :

$$P \Rightarrow_{\text{R-M}} Q = \begin{cases} \mathbf{1} & \text{si } p \leq q \\ \boldsymbol{q} & \text{si } p > q \text{ et } p = 1 \\ \mathbf{1}^- & \text{si } p > q \text{ et } p < 1 \end{cases}$$

où 1^- désigne une valeur limite strictement inférieure à 1 (le comportement discontinu de cette implication provient de celui de la norme triangulaire sous-jacente).

Quelques caractéristiques des implications floues

Les implications floues présentées précédemment généralisent toutes l'implication usuelle (appelée aussi **implication matérielle**), en ce sens que leur résultat coïncide avec celui de l'implication usuelle quand les valeurs en entrée sont *vrai* et/ou *faux*. En effet, pour les S-implication, 1 étant un élément absorbant et 0 élément neutre de toute co-norme, on a :

- (faux $\Rightarrow_{\text{S-i}}$ faux) = $\perp(1, 0) = 1$,
- (vrai $\Rightarrow_{\text{S-i}}$ vrai) = $\perp(0, 1) = 1$,
- (vrai $\Rightarrow_{\text{S-i}}$ faux) = $\perp(0, 0) = 0$,
- (faux $\Rightarrow_{\text{S-i}}$ vrai) = $\perp(1, 1) = 1$.

De façon analogue, pour les R-implications et leurs contraposées :

- (faux $\Rightarrow_{\text{R-i}}$ faux) = (faux $\Rightarrow_{\text{R-i}}$ vrai) = vrai $\Rightarrow_{\text{R-i}}$ vrai) = 1 puisque le premier argument est inférieur ou égal au second,
- (vrai $\Rightarrow_{\text{R-i}}$ faux) = $\sup_{[0, 1]} \{u \mid T(1, u) \leq 0\} = \sup_{[0, 1]} \{u \mid u \leq 0\} = 0$.
  Le fait que l'implication de Rescher-Gaine satisfait cette propriété est aisément vérifiable.

Les deux classes d'implications floues considérées auparavant possèdent les propriétés suivantes caractérisant les implications floues selon **[YAG 80]**

- $p \Rightarrow_{\text{f}}$ vrai = 1,
- faux $\Rightarrow_{\text{f}} q = 1$,
- vrai $\Rightarrow_{\text{f}} q = q$,
- si $q > r$ alors $(P \Rightarrow_{\text{f}} Q) \geq (P \Rightarrow_{\text{f}} R)$,
- si $p < r$ alors $(P \Rightarrow_{\text{f}} Q) \geq (R \Rightarrow_{\text{f}} Q)$.

Sémantique associée aux implications floues

Au-delà de ces aspects, il apparaît particulièrement important de s'intéresser à la signification qui peut être attribuée aux diverses implications floues. Ceci permet en particulier de procéder à un choix approprié et d'en expliciter le sens. Les S-implications (Kleene-Dienes, Reichenbach et Lukasiewicz) présentent la caractéristique de garantir un *niveau de satisfaction minimal* de (1- *p*). De plus, pour l'implication de Kleene-Dienes, *p* joue le rôle d'un niveau d'importance attribué à la conclusion *Q*. En effet, si *p* est faible, on "voit" seulement les valeurs élevées de *Q*. En effet, si *p* est faible, on "voit" seulement les valeurs élevées de *Q,* contrairement au cas où *p* est élevé pour lequel *Q* est "vue" sur une plage de valeurs large (1- *p* est faible quand *p* est grand et donc l'intervalle [1- *p*, 1] sur lequel on "voit" *Q* est alors grand).

Pour ce qui est des R-implications, la valeur de la prémisse *P* joue le rôle d'un seuil qui conduit à la totale satisfaction quand il est atteint ou dépassé par la valeur de la conclusion *Q* et à une satisfaction partielle sinon. Dans ce dernier cas, le degré obtenu *ne dépend que de la conclusion* avec l'implication de Gödel, varie avec le *ratio* entre la valeur de la conclusion et celle de la prémisse avec l'implication de Goguen et avec leur *écart* (ou distance) avec l'implication de Lukasiewicz.

IV.2- La théorie des possibilités

La **théorie des possibilités** a été introduite en 1978, toujours par Zadeh **[ZAD 78]** puis développée notamment dans **[DUB 80][DUB 88].** Cette théorie permet de raisonner sur des connaissances *imprécises* en introduisant un moyen de prendre en compte *l'incertitude* de ces connaissances. La théorie des possibilités permet de traiter à la fois les informations imprécises et incertaines.

La distinction entre imprécision et incertitude a été particulièrement évoquée par Dubois et Prade dans **[DUB 88b]**.

A)- Incertitude

L'incertitude se réfère au manque d'information disponible sur l'état du monde pour déterminer si une affirmation classique (qui peut être seulement vraie ou fausse) est actuellement vraie ou fausse. Une information est incertaine dès qu'on ne peut l'affirmer ou la nier avec certitude totale. L'incertitude est relative à la confiance que l'on accorde à l'information (source de l'information).

B)- Imprécision

Une information est imprécise dès que l'ensemble des valeurs de la variable qu'elle spécifie n'est pas réduit à un singleton. L'imprécision se réfère au contenu de l'affirmation considérée et dépend de la granularité du langage utilisé pour décrire l'information (l'imprécision est une notion qui dépend du contexte. Ceci se traduit par le fait qu'une information qui est précise dans un contexte, peut ne plus l'être dans un autre contexte. Par exemple : "Omar a entre 25 et 30", cette phrase est clairement imprécise ; mais la phrase "Omar a 28 ans" est précise si on attribut seulement les valeurs pour les années et imprécise si la valeur devait indiquer aussi le nombre de mois.).

L'imprécision se traduit soit par une hésitation face à plusieurs choix entre lesquels on ne peut pas trancher, soit par des valeurs mutuellement exclusives mais dont aucune n'est exclue **[DUB 94]**.

L'imprécision peut provenir par exemple :

- d'une mauvaise connaissance des informations numériques qui peut être la conséquence d'une insuffisance des instruments d'observation (4000 à 5000 touristes), d'erreurs de mesures (poids à 1% près) ou encore d'une connaissance approximative (le prix d'un bon ordinateur familial est environ entre 800 et 1100 euros),
- d'un contenu informationnel représenté par des connaissances approximatives et vagues en termes de langage naturel,
- de sources d'information distinctes utilisant des vocabulaires différents,
- les valeurs imprécises peuvent aussi être vues comme des raffinements de valeurs manquantes. Les valeurs imprécises peuvent être obtenues à partir de données expérimentales **[MAS 06]**, et remplacent donc les valeurs manquantes.

Remarques :

- La théorie des probabilités représente un cadre efficace pour traiter l'incertain,
- La théorie des ensembles flous représente un cadre efficace pour traiter l'imprécis,
- Imprécision et incertitude sont parfois liées. En effet, une donnée imprécise confère une incertitude quant à l'exploitation (calcul, décision, etc.) qu'on peut faire de cette donnée. Par exemple l'information « la température de la pièce est d'environ 16° » est imprécise. Mais le calcul de la pression atmosphérique serait incertain.

IV.2.1- Mesures de possibilité d'événement usuel

Dans ce cadre, l'incertitude d'un événement E est représentée à la fois par la mesure de possibilité de cet événement et par la mesure de possibilité de l'événement contraire, ces degrés étant "faiblement" liés. La notion de mesure de possibilité est présentée dans cette section.

La mesure de *possibilité* Π d'un événement E défini sur un ensemble de référence X est une valeur, comprise entre 0 et 1, évaluant à quel point cet événement $E \subseteq X$ est possible.

Définition 6 : (Mesure de possibilité)

Soient X un référentiel et E_i (i=1..n) des sous-ensembles de X. Une mesure de possibilité est une fonction ensembliste notée Π définie par : $P(X) \rightarrow [0, 1]$, $P(X)$ étant l'ensemble des parties de X, qui vérifie les trois (03) axiomes suivants (au sens de la théorie de la mesure) :

1) $\Pi(X) = 1$,
2) $\Pi(\varnothing) = 0$,
3) $\Pi(\cup_{i=1..n} E_i) = \mathbf{Max}_{i=1..n}(\Pi(E_i))$.

A partir des trois axiomes **1), 2)** et **3)** un ensemble de propriétés peuvent être déduites.

IV.2.1.1- Propriétés des mesures de possibilités

- Soit E un ensemble usuel, alors $E \cup E^c = X$. Le premier et le troisième axiome permettent d'obtenir la propriété : $\mathbf{Max(\Pi(E), \Pi(E^c)) = 1}$. Cette équation traduit le fait que, de deux événements contraires, l'un au moins est possible : $\mathbf{\Pi(E) < 1 \Rightarrow \Pi(E^c) = 1.}$
- D'autre part, la possibilité d'un événement n'implique pas l'impossibilité de son événement contraire. En clair, de deux événements contraires, l'un au moins est complètement possible,
- $\Pi(E) = 1$ (mesure de possibilité normalisée) signifie que l'événement E est complètement possible. De manière générale, plus $\Pi(E)$ se rapproche de 1, plus l'événement E est possible,
- Au-delà de l'axiomatisation de la mesure de possibilité qui concerne la disjonction de deux événements E et F, il existe une relation entre la possibilité de la conjonction des événements E et F et leurs possibilités individuelles, à savoir : $\mathbf{\Pi(E \cap F) \leq Min(\Pi(E), \Pi(F))}$. Dans le cas particulier où les événements E et F sont indépendants, on a : $\mathbf{\Pi(E \cap F) = Min(\Pi(E), \Pi(F))}$.

IV.2.1.2- Distribution de possibilité

Une distribution de possibilité π représente les valeurs de possibilité sur un ensemble X fini. π est une fonction définie par $\pi : X \rightarrow [0, 1]$ tel que $\exists\, x \in X\, ; \pi(x) = 1$.

Une mesure de possibilité Π peut être obtenue à partir d'une distribution de possibilité par la relation suivante :

$$\forall E \subseteq X ;\ \mathbf{\Pi(E) = \sup_{x \in E}\ \pi(x)} \qquad \textbf{(IV.23)}$$

- Le nombre $\Pi(E)$ quantifie dans quelle mesure l'événement $E \subseteq X$ est possible.

Représentation d'une distribution de possibilité par une α-coupe

Une distribution de possibilité peut être aussi définie par l'ensemble de ses α-coupes π_α définies par $\forall \alpha \in]0, 1] : \pi_\alpha = \{x \in X \mid \pi_\alpha(x) \geq \alpha\}$

- On a $0 < \alpha \leq \beta \leq 1 \Rightarrow \pi_\alpha \supseteq \pi_\beta$.. Ceci permet d'écrire π_α comme :

$$\mathbf{\forall\, x\ ;\ \pi_\alpha(x) = Sup\{\alpha \mid x \in \pi_\alpha\}} \qquad \textbf{(IV.24)}$$

- La coupe au niveau $\alpha = 1$ de π est dite noyau : $\pi_1 = \{x \in X \mid \pi(x) = 1\}$
- La coupe au niveau $\alpha = 0$ de π est dite support : $Supp(\pi) = \{x \in X \mid \pi(x) > 0\}$

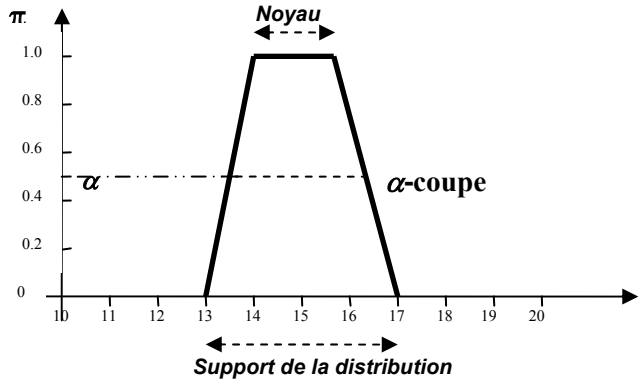


Fig. IV.5. Exemple de distribution de possibilité linéaire par morceaux

IV.2.1.3- Cas extrêmes Ignorance totale et Certitude totale

Il existe deux cas extrêmes de la réalisation d'un événement E qui correspondent aux situations d'*Ignorance totale* et *certitude totale*. Ces deux cas sont définis comme suit :

Définition 7 : (Ignorance totale)

Cette situation correspond au cas particulier où E et E^c ont même degré de possibilité, ($\Pi(E) = 1$ et $\Pi(E^c) = 1$). Cela est dérivé à partir de la propriété $\text{Max}(\Pi(E), \Pi(E^c)) = 1$), les deux événements contraires sont tous deux complètement possibles,

Définition 8 : (Certitude totale)

La certitude totale de l'événement E correspond à la situation où la réalisation de E est complètement certaine, ce qui signifie que l'événement E^c est complètement impossible. La certitude totale est représentée par : $\Pi(E) = 1$ et $\Pi(E^c) = 0$.

Dans ces deux situations extrêmes (ignorance totale et certitude totale), la valeur de $\Pi(E) = 1$, ce qui montre que l'information apportée par la mesure de possibilité doit être complétée par une autre information capturée par la notion de *nécessité* (ou certitude).

IV.2.2- Mesure de nécessité d'un événement usuel

La mesure de *nécessité* N d'un événement défini sur X est une valeur, comprise entre 0 et 1, quantifiant jusqu'à quel point l'événement est certain. Elle est définie à partir de l'impossibilité de l'événement contraire. Formellement :

$$\mathbf{N(E) = 1 - \Pi(E^c)} \qquad \textbf{(IV.25)}$$

Définition 9 : (Mesures de nécessité)

Soient X un référentiel et E_i (i=1..n) des sous-ensembles de X. Une mesure de nécessité est une fonction ensembliste, notée N, définie de $P(X)$ dans [0, 1] qui vérifie les trois (03) axiomes suivants :

1. $\mathbf{N(\emptyset) = 0}$,
2. $\mathbf{N(X) = 1}$,
3. $\mathbf{N(\cap_{i=1,2...} E_i) = Min_{i=1,2...}(N(E_i))}$.

La définition d'une mesure de Nécessité à partir de la distribution de possibilité π, est donnée par la formule suivante :

$$\mathbf{\forall\, E \in P(X),\ N(E) = \inf_{x \in E}(1 - \pi(x))} \qquad \textbf{(IV.26)}$$

Remarque :

Avant d'être nécessaire (certain), un événement doit être complètement possible.

A partir des trois axiomes **1., 2.** et **3.,** un ensemble de propriétés de la mesure de nécessité peuvent découler. Elle sont résumées dans ce qui suit.

Propriétés des mesures de nécessité

- La valeur N(E) indique le degré de certitude que l'événement E soit réalisé. Plus la valeur N(E) se rapproche de 1 et plus il est certain que E soit réalisé,
- Les mesures de nécessité satisfont la relation suivante : **Min($N(E)$, $N(E^c)$) = 0**, d'où **N(E) ≤ Π(E)**,
- Si N(E) = 1 alors il est complètement certain que E se réalise (donc $\Pi(E^c) = 0$, ce qui implique que $\Pi(E) = 1$),
- $N(E) > 0 \Rightarrow \Pi(E) = 1$. Elle signifie que si un événement est un temps soit peu certain alors il est complètement possible,
- Comme $E \cup E^c = X$, la relation suivante est valide : $\Pi(E) < 1 \Rightarrow N(E) = 0$. Elle signifie que si l'événement E n'est pas totalement possible, alors il est exclu qu'il soit un tant soit peu certain.

IV.2.3- Possibilité et Probabilité

La notion de possibilité repose sur une fonction ensembliste qui, de manière analogue aux probabilités, évalue la réalisation d'événements. Cependant, les mesures de possibilité et de probabilité véhiculent deux sémantiques bien distinctes.

Une *valeur de probabilité* fournit une fréquence de la réalisation d'un événement, ce qui permet par ailleurs d'ordonner les différents événements selon leur fréquence.

Une *valeur de possibilité* est purement ordinale au sens qu'elle ne vise qu'à ordonner les différents choix.

Par exemple, si l'univers X= {rouge, noir} représente les résultats d'une roulette de casino, la probabilité p définie par p({rouge}) = 0.8 et p({noir}) = 0.2 indique que la fréquence de l'événement «le rouge (resp. noir) sort » est de 8 (resp. 2) fois sur 10. L'événement « le rouge sort » est 4 fois plus fréquent que l'événement « le noir sort ». Si on doit miser sur l'une des deux couleurs, le rouge sera classé en premier.

Si le résultat d de la roulette est modélisé par des possibilités, il est toujours possible de classer les deux éventualités mais les deux fréquences des événements ne seront pas prises en considération pour ce classement.

D'une manière générale, la théorie des possibilités est adaptée au contexte où des fréquences ne sont pas disponibles et où une estimation subjective intervient.

IV.2.4- Possibilités et ensemble flou

Il existe un lien entre la théorie des possibilités et celle des ensembles flous car une distribution de possibilité peut être représentée par un ensemble flou normalisé dont les éléments sont les événements élémentaires. Il est alors important de comprendre qu'un ensemble flou ne représente plus une conjonction d'éléments (au sens habituel des ensembles) mais une disjonction d'événements élémentaires et qu'en conséquence il ne peut être manipulé avec les opérateurs ensemblistes de l'algèbre ensembliste vus précédemment.

Exemple 2 : de distribution de possibilité

La distribution de possibilité de « l'âge d'une personne compris entre 20 et 30 ans » peut être représenté par l'ensemble flou normalisé : {0.2/20+ 0.4/22+ 0.6/23+ 0.8/24+ 1/25 + 0.8/26+ 0.6/27+ 0.4/28+ 0.3/29+ 0.2/30}

IV.2.5- Possibilité/ Nécessité d'événements flous

Etant donné un ensemble de référence X, la fonction d'appartenance μ_E du sous ensemble flou E peut être vue comme la distribution de possibilité π, c'est-à-dire que, si t est le degré d'appartenance de x à E, t peut être interprété comme le degré de possibilité pour que X soit égal à x. Etant donnée cette distribution de possibilité, la possibilité et la nécessité d'un événement flou E sont définis dans ce qui suit.

IV.2.5.1- Possibilité d'événements flous

La notion de possibilité d'événement flou a été introduite par Zadeh **[ZAD 78].** A partir de la distribution de possibilité π (définie sur l'univers X), la mesure de possibilité de l'événement flou E (d'univers X) est définie par :

$$\Pi(E) = \mathrm{Sup}_{u \in X}\, \mathrm{Min}(\mu_E(u), \pi(u)) \qquad \textbf{(IV.27)}$$

Cette définition retient le cas le plus favorable (via le supremum sur les éléments u de X), chaque cas exprimant la conjonction (sémantique du minimum) : « l'événement élémentaire u est conforme à E ($\mu_E(u)$) et u est possible ($\pi(u)$) ».

IV.2.5.2- Nécessité d'événement flou

La nécessité d'un événement flou est définie de façon analogue à celle d'un événement ordinaire par :

$$\mathbf{N(E) = 1 - \Pi(E^c) = 1 - Sup_{u \in X}\ Min(1-\mu_E(u), \pi(u))} \qquad \mathbf{(IV.28)}$$

$$\mathbf{= Inf}_{u \in X}\ \mathbf{Max(\mu_E(u), 1 - \pi(u))}$$

Si la valeur de N(E) vaut 1, pour toute éventualité u, soit la valeur $\mu_E(u)$ vaut 1, soit la valeur $1-\pi(u)$ vaut 1, ou en d'autres termes, « pour tout u, si u est un tant soit peu possible alors u est complètement conforme à E », ce qui signifie que la distribution de possibilité est incluse dans le noyau de l'ensemble flou décrivant E. plus généralement, la valeur de N(E) peut être considérée comme un degré exprimant dans quelle mesure toute valeur plus ou moins possible est en adéquation avec la condition floue.

IV.2.5.3- Propriétés de mesures de possibilité/nécessité floues

- L'expression de N(E) peut se réécrire en utilisant l'implication floue de Kleene-Dienes ($X \Rightarrow_{K\text{-}D} Y = \text{Max}(1- X, Y)$), ce qui donne un indice d'inclusion :

  $N(E) = \inf_{u \in X} \pi(u) \Rightarrow_{K\text{-}D} \mu_E(u)$

- la propriété $E \cup E^c = X$ n'étant généralement pas vérifiée pour les ensembles flous (sauf avec la co-norme de Lukasiewicz par exemple), la relation : $\Pi(E) < 1 \Rightarrow N(E) = 0$ n'est plus valide. Elle est remplacée par la relation plus générale (en ce sens qu'elle est également vraie pour les événements ordinaires) : $\Pi(E) \geq N(E)$. Ainsi, tout événement ne peut être plus certain qu'il n'est possible.

IV.2.6- Intérêt de la théorie des possibilités

La théorie des possibilités présente plusieurs intérêts pour la manipulation d'informations incomplètes. Les plus importants sont :

- La prise en compte combinée de l'imprécision et de l'incertitude dans des connaissances,
- L'expression dans quelle mesure la réalisation d'un évènement est possible et dans quelle mesure on en est certain, sans toutefois avoir à sa disposition l'évaluation de la probabilité de cette réalisation,
- Le raisonnement sur des connaissances imprécises et/ou vagues.

L'incomplétude des informations en générale, et l'imprécision en particulier, est un phénomène présent dans les bases de données et qui touche les valeurs. Les valeurs imprécises peuvent apparaître dans de nombreuses situations. On peut citer à titre d'exemples :

- La constitution d'entrepôts de données ou plus généralement la fusion de données dont les sources sont plus ou moins fiables apportent des informations non identiques,
- Les bases contenant des données obtenues par des mécanismes de reconnaissance automatique délivrant des résultats intrinsèquement imprécis (reconnaissance d'images ou d'objets).

Les sections suivantes de ce chapitre seront consacrées à présenter les différentes approches utilisées pour étendre le modèle relationnel et la notion de dépendances fonctionnelles classiques en utilisant la théorie des ensembles flous et celle des possibilités.

IV.3- Les principaux modèles de bases de données floues

Plusieurs modèles sont apparus dans la littérature pour étendre le modèle relationnel aux données floues. Nous pouvons citer : le modèle de Pons **[PON 95],** le modèle de Buckles-Petry **[BUC 82],** le modèle d'Umano-Fukami **[UMA 80],** le modèle de Prade-Testemale **[PRA 82],** le modèle GEFRED de Medina-Pons-Vila **[MED 94]** et le Genralized Possibilistic Relational Model **[DJO 07]**. Nous présentons dans cette section ces modèles en soulignant leurs avantages et leurs limites.

IV.3.1- Modèle de Pons

Le modèle de Pons se voit d'une forme très simple. Il consiste à ajouter un degré de vérité, habituellement dans l'intervalle [0, 1], à chaque tuple. Ceci autorise à maintenir l'homogénéité des données dans la BD. Cependant, la sémantique qui est assignée à ce degré sera celle qui détermine son utilité et, par conséquent, cette sémantique sera utilisée dans les processus de consultation.

Plus formellement, une BD floue (BD = $\{R_1, R_2, ..., R_n\}$) sera un ensemble de relations floues. Chaque relation est caractérisée par une fonction d'appartenance μ_{Ri} **[PON 95]** tel que :

$$\mu_{Ri} : U_1 \times U_2 \times \ldots \times U_i \times \ldots \times U_m \rightarrow [0, 1] \qquad \textbf{(IV.29)}$$

Avec U_i est le domaine de l'*i*-ème attribut de la relation et × est le produit Cartésien.

Les principaux inconvénients de ce modèle sont :

- Il ne permet pas de représenter l'imprécision au niveau attribut,
- Il assigne à chaque tuple un caractère flou de forme globale sans déterminer quelle est la participation de chaque attribut.

IV.3.2- Modèle de Buckles-Petry

Ce modèle, proposé par Buckles et Petry **[BUC 82],** utilise les relations de similitude**.** Une relation floue est définie comme un sous-ensemble du produit Cartésien $2^{D1} \times 2^{D2} \times \ldots \times 2^{Dn}$.
Où 2^{Di} désigne l'ensemble des parties du domaine ($P(D_i)$).

Les valeurs qui peuvent être représentées sont les suivantes :

- Ensemble fini de scalaires. Exemple {Blond, marron, roux},
- Ensemble fini de nombre. Exemple {10, 11, 12},
- Ensemble d'étiquettes floues (variables linguistiques). Exemple {petit, moyen, grand}.

La relation de ressemblance qui existe sur chacun des domaines sert à représenter et diriger l'imprécision. Elle établit une mesure de similitude entre les différentes valeurs du domaine sur lequel elle est définie. Elle est définie par l'utilisateur et les valeurs de ressemblance sont comprises entre 0 et 1. (0 : Complètement différent ; 1 : Complètement semblable).

Dans une consultation (interrogation), l'utilisateur pose une question sur les tuples satisfaisants une condition déterminée pour un seuil de similitude donné. Les tuples de la relation se regroupent en classes d'équivalence, suivant les relations et les seuils de similitudes définis sur eux.

Exemple 3 :

Soit la relation *Personne* décrite dans la table suivante et soit l'attribut *couleur_de_cheveux* défini sur le domaine $D_{couleur\_de\_cheveux}$ = {blond, châtain, roux, noir}.

Extension de la relation *Personne*

nom	Age	Taille	Couleur_de_cheveux
Med Ali	16	{grand, très grand}	Châtain
Ramzi	17	Petit	Noir
Sami	15	Très petit	Blond
Sana	{15, 16}	moyen	Roux

Supposons que nous avons les relations de similitude suivantes :

Relations de similitude

	Blond	Châtain	Roux	Noir
Blond	1	0.6	0.4	0
Châtain	0.6	1	0.5	0.1
Roux	0.4	0.5	1	0.8
Noir	0	0.1	0.8	1

Soit la requête suivante : "*donner les noms et les tailles des personnes ayant la couleur des cheveux semblable au châtain avec un degré* 0.6". Dans cette requête, le seuil de similitude est seuil ($D_{couleur\_de\_cheveux}$) = 0.6 et la classe d'équivalence est donnée par :

Définition des classes d'équivalences

Nom	Age	Taille	Couleur-de-cheveux
{Med Ali, Sami}	{16, 15}	{très grand, grand, Très petit}	{châtain, Blond}
Ramzi	17	petit	noir
Sana	{15, 16}	Moyen	Roux

Le résultat final (après l'application des opérateurs relationnels) est :

Résultat final de la requête

Nom	Hauteur
{Med Ali, Sami}	{très grand, grand, Très petit}

Les principaux inconvénients de ce modèle sont :

- Il ne modélise pas bien tous les aspects flous de l'information,
- Il ne garantit pas l'atomicité dans la représentation de l'information,
- Il peut donner lieu à des résultats confus lors de l'application des opérateurs relationnels aux classes d'équivalences,
- Il présente un résultat possédant plusieurs interprétations.

Cependant, nous pouvons signaler une série d'avantages :

- L'emploi de relations de ressemblance fournit un outil approprié et intuitif pour représenter l'imprécision des concepts,
- L'utilisation de différents seuils pour chacun des attributs impliqués dans une consultation.

IV.3.3- Modèle d'Umano-Fukami

Ce modèle se base sur la théorie des possibilités **[UMA 80]**. La différence avec les autres modèles réside dans la manière de décrire l'information floue. Il utilise comme valeurs :

- Les distributions de possibilité,
- Les valeurs indéterminées, inconnues et nulles avec les sémantiques suivantes :
  - Undefined: $\pi_{A(x)}(d) = 0, \forall\ d \in D$,
  - Unknown: $\pi_{A(x)}(d) = 1, \forall\ d \in D$,
  - Null : $\pi_{A(x)}(d) = \{1/\text{Unknown}, 1/\text{Undefined}\}\ \forall\ d \in D$.

Avec D est l'univers de discours de $A(x)$ et $\pi_{A(x)}(d)$ représente la possibilité que $A(x)$ prenne la valeur d de D.

Le résultat de la consultation retourne trois sous-ensembles disjoints :

- Les tuples qui satisfont clairement la consultation,
- Les tuples qui satisfont approximativement la consultation,
- Les tuples qui ne satisfont pas clairement la consultation.

L'avantage de ce modèle est qu'il peut assigner des distributions de possibilité au valeurs des tuples, comme il peut assigner un degré d'appartenance à chaque tuple de la relation pour une requête donnée (satisfaisant, non satisfaisant, possiblement satisfaisant).

Exemple 4 :

Soit la relation « *Personne* » donnée par l'extension suivante :

Nom	**Age**	**Enfants**
Med Ali	23	Yacine
Amel	35	youcef
Kamel	Jeune	Unknown
Tarek	Unknown	Undefined
Souad	{50, 51}	Null

Supposons que la distribution de possibilité de la variable linguistique floue "*jeune*" soit jeune = 0.3/15+ 0.6/16+0.8/17+1/19+1/20+1/21+1/22+1/23+0.9/25+0.8/25+0.7/26. Si nous établissons une requête de type : "donner les personnes qui ont plus de 25 ans", nous aboutissons au résultat comprenant les trois sous-ensembles suivants :

– Clairement satisfaisant = 1/Amel+1/Hider+1/Souad,

– Possiblement satisfaisant = 1/Kamel+1/Tarek,

– Clairement non satisfaisant = 1/Med Ali.

IV.3.4- Modèle de Prade-Testemale

Ce modèle se base sur la théorie des possibilités pour modéliser les données imprécises et/ou incomplètes **[PRA 82]**. Sa structure des données est similaire à celle utilisée dans le modèle d'Umano-Fukami. Il utilise des mesures de possibilité et de nécessité pour la satisfaction des conditions établies dans la consultation.

Les valeurs du domaine D_j d'un attribut A_j dans ce modèle est soit :

- Une unique valeur $d \in D_j$ parfaitement connu. Exemple âge(Ramzi) = 19 ou *couleur_cheveux*(Med) = noir,
- Une valeur nulle (Null) qui n'inclut pas la valeur inconnue (Unknown) et non applicable (Undefined),
- Une variable linguistique. Exemple, Taille(Ramzi) = "pas très haut", où "pas très haut" est une distribution de possibilité sur le domaine de l'attribut « taille », de la forme : 0.8/1m60+ 0.9/1m65+ 1/1m70+ 1/1m75+ 0.8/1m80.

Exemple 5 :

Soit la relation *Etudiant*(Nom, Age, Mathematiques, Informatique) :

Dans l'extension de cette relation, l'intervalle [8,10] représente un intervalle de valeurs, alors que "*bien*", "*jeune*", etc., représentent des distributions de possibilité. Pour répondre à une requête de type "*trouver les étudiants dont la note en informatique est mieux que moyen*", il faut définir la distribution de possibilité qui correspond à "*moyen*" et le comparateur flou "*mieux que*".

Extension de la relation *Etudiant*

Nom	Age	Mathématiques	Informatique
Amel	Jeune	8	[8, 10]
Hider	Approximativement 24	7	Undefined
Med Ali	[22, 24]	Légèrement mal	Pas très mal
Ramzi	19	8	9.5
Kamel	Unknown	Bien	Bien
Nourdine	Très-jeune	9	Approximativement 5
Sami	22	3	7

Soit : *moyen* ={0.12/1, 0.2/6, 0.4/7, 0.7/8, 0.95/9.5, 1/10, 0.3/13.5, 0.25/14, 0.12/14.5}. Et soit le comparateur flou "*mieux que*" défini par :

$$\mu_{mieux\_que}(u, v) = \begin{cases} 0 & \text{si } u - v \leq 2 \\ 0.5 & \text{si } u - v = 3 \\ 1 & \text{si } u - v \geq 4 \end{cases}$$

Résultat de la requête :

Nom	Age	Mathématiques	Informatique	$\mu_{mieux\_que}(u,moyen)$
Kamel	Unknown	Bien	Bien	0.2

Avantages :

- Les auteurs ont proposé une extension de l'algèbre relationnelle à l'algèbre floue avec jointure floue, restriction floue, etc.

IV.3.5- Modèle GEFRED de Médina et al

Le modèle GEFRED (GEneralised model for Fuzzy RElationnel Database) a été proposé en 1994 par Medina, Pons et Vila **[MED 94]**. Il constitue une synthèse des différents modèles

publiés pour traiter le problème de la représentation et du traitement des informations floues au moyen des BDR. Un des principaux avantages de ce modèle est qu'il consiste à une abstraction générale qui permet de traiter différentes approches, même celles qui peuvent paraître très disparates. Il se base sur le Domaine Flou Généralisé (*D*) et sur la Relation Floue Généralisée (*R*), qui incluent respectivement les domaines et les relations classiques. Nous présentons dans ce qui suit les concepts de base de ce modèle.

Définition 10 : (Domaine Flou Généralisé)

Si *U* est le domaine du discours, *P*(*U*) est l'ensemble de toutes les distributions de possibilité défini sur *U*, y compris celles qui définissent les types Inconnu et Indéterminé. Le Domaine Flou Généralisé, est défini comme $D \subseteq P(U) \cup \text{NULL}$.

Le Domaine Flou Généralisé constitue l'élément structurel qui organise la représentation des types de données **[MED 94a]**.

Définition 11 : (Relation Floue Généralisée)

Une Relation Floue Généralisée, *R*, est donné par deux ensembles "Entête" (Head H) et "Corps" (Body B). *R* = (H, B), définie par :

- **L'Entête** : elle comporte un ensemble fixe de triplet "attribut-domaine-compatibilité" (en anglais "attribute-domain-compatibility") où le dernier est optionnel,

$$\mathbf{H} = \{(A_1 : D_1[,C_1]), (A_2 : D_2[,C_2]), ..., (A_n : D_n[,C_n])\}$$

Où C_j est un "attribut de compatibilité" qui prend des valeurs dans l'intervalle [0, 1].

Types de données dans GEFRED

Les données dans ce modèles sont soit :

a. Une seule valeur linguistique (comportement = bon, représenté par la distribution de possibilité, 1/bon),
b. Un seul nombre (Âge = 28, représenté par la distribution de possibilité, 1/28),
c. Un ensemble de distributions linguistiques mutuellement exclusives (comportement = {bon, mauvais}, représenté par {1/bon, 1/mauvais}),
d. Un ensemble de distributions de possibilité numériques possibles mutuellement exclusives (Âge = {20, 21}, représenté par {1/20, 1/21}

e. Une distribution de possibilité dans le domaine linguistique {comportement = {0.6/mauvais, 0.7/normal}).

f. Une distribution de possibilité dans le domaine numérique (Âge = {0.4/23, 1.0/24, 0.8/25}, nombres flous ou étiquettes linguistiques).

g. Un nombre réel qui appartient à [0, 1] référencié à un degré de réalisation (Qualité = 0.9).

h. Une valeur inconnue (UNKNOWN) avec une distribution de possibilité, UNKNOWN = $\{1/u : u \in U\}$.

i. Une valeur indéterminée (UNDEFINED) avec une distribution de possibilité, UNDEFINED = $\{0/u : u \in U\}$.

j. Une valeur nulle (NULL) donnée par NULL = {1/ Undefined, 1/Unknown}.

- **Le Corps** : il comporte un ensemble de tuples, appelés "tuples flous généralisés"(en anglais generalized fuzzy tuples), où chaque tuple est composé d'un ensemble de triplet "attribut-valeur-degré" (attribute-value-degrees), où le degré est optionnel.

$$\mathbf{B} = \{(A_1 : d_{i1}[, c_{i1}]), (A_2 : d_{i2}[, c_{i2}]), ..., (A_n : d_{in}[, c_{in}])\}$$

Avec ($i = 1..m$), et m est le nombre de tuples dans la relation et où le d_{ij} représente la valeur du domaine pour le tuple t_i et l'attribut A_j , et le c_{ij} est le degré de la compatibilité associé à cette valeur **[MED 94]**.

Le degré de compatibilité est utilisé pour savoir le degré avec lequel une valeur d'un attribut a satisfait la condition de la requête.

IV.3.6- Le Modèle Relationnel Possibiliste Généralisé

Le Modèle Possibiliste Généralisé (en anglais Generalized Possibilistic Relational Model) **[RED 06][DJO 07]** a été proposé afin d'étendre l'imprécision au schéma même de la base données de données floues. Autrement dit, tous les modèles de base données de données floues présentées jusqu'ici (**§IV.3.1** à **§IV.3.2**) considéraient que seulement l'extension $\tilde{r}$ d'une relation $\boldsymbol{R}(A_1, A_2, \ldots, A_n)$ contenait des valeurs floues. Certaines recherches ont montré qu'il était parfois nécessaire de considérer un ou plusieurs attributs A_i comme étant lui-même

un ensemble flou. Ce qui n'est pas le cas pour les approches présentées précédemment. Le schéma de la relation considéré est donc $\tilde{\boldsymbol{R}}(\tilde{A}_{1,1},\ldots,\tilde{A}_{1,h1},\ldots,\tilde{A}_{n,1},\ldots,\tilde{A}_{n,hn})$.

L'extension $\aleph$ d'une relation possibiliste $\tilde{\boldsymbol{R}}(\tilde{A}_{1,1},\ldots,\tilde{A}_{1,h1},\ldots,\tilde{A}_{n,1},\ldots,\tilde{A}_{n,hn})$ généralisé est obtenue en utilisant un opérateur de restriction β sur les collections de distribution de possibilité. Une telle extension est définie comme suit :

$$\aleph \subseteq (\zeta(D_1)\ \beta\ \tilde{A}_{1,1} \times \ldots \times (\zeta(D_j)\ \beta\ \tilde{A}_{1,kj}) \times \ldots \times (\zeta(D_n)\beta\ \tilde{A}_{n,hn}))$$

- Les notations utilisées dans ce modèle sont :

  - $\tilde{\boldsymbol{A}}_{\boldsymbol{j,k}}$: le $k^{\text{ème}}$ ensemble flou (intervalle fou) de l'attribut A_j, $1 \leq k \leq h_j$.
  - $\boldsymbol{D_j}$: le domaine des valeurs précises d'un attribut A_j.
  - $\boldsymbol{v_{ij}}$: une valeur précise de l'attribut A_j dans le tuple t_i (i=1..n, j= 1..m),
  - $\boldsymbol{\pi_{ij}}$: la distribution de possibilité (valeur imprécise) de l'attribut A_j dans le tuple t_i.
  - $\boldsymbol{\zeta(D_j)}$: la collection des distributions de possibilité de l'attribut A_j définie sur son domaine D_j et $\times$ représente le produit Cartésien,
  - $\boldsymbol{\mu_{\tilde{A}jk}}$: la fonction d'appartenance de l'ensemble flou $\tilde{A}_{jk}$ ($\mu_{\tilde{A}jk} : D_j \rightarrow [0, 1]$).
  - β : l'opérateur de restriction flou. Cet opérateur est défini dans la sous-section suivante.

Sémantique de l'opérateur de restriction flou

Sémantiquement, une bases de données possibiliste est interprétée comme étant un ensemble de bases de données possibles (appelées mondes possibles). Chaque monde possible est obtenu en choisissant une valeur candidate appartenant à la distribution de possibilité **[BOS 03].** Pour une base de données possibiliste contenant « 3 » valeurs imprécises représentées par des distributions de possibilité, où chaque valeur contient au maximum 4 valeurs candidates ; le nombre de mondes possibles est de 3×4. Ce type d'interprétations génère un nombre important de représentations pour des bases de données contenant un nombre important de distributions de possibilité.

Réciproquement à cette interprétation inflationniste, on suggère une interprétation restrictive basée sur la restriction des valeurs possibles sur un ensemble flou. Un opérateur de restriction β est donc utilisé pour caractériser la sémantique des bases de données possibilistes généralisées donnée si dessous.

Supposons la restriction floue (ensemble flou) F, avec son degré d'appartenance μ_F. L'ensemble des modèles [F] de F est lui-même flou. Supposons aussi que le cas épistémique de réalisation $A(X)$ est représenté par une distribution de possibilité π définie sur le domaine de F (notée par Δ). Puisque les valeurs précises de $A(X)$ ne sont pas connues, le fait que $A(X)$ satisfait ou pas la restriction floue F peut être incertain. L'évaluation de la possibilité (resp. la nécessité) de réalisation de $A(X)$ par rapport à la restriction floue F est donnée par l'ensemble flou ΠF (resp. NF). De ce fait, deux sémantiques sont associées à l'opérateur de restriction ($\pi_{A(X)}(d)$ β $\mu_F(d)$). Ces sémantiques sont représentées par les ensembles flous ΠF et NF dont les fonctions d'appartenance sont décrites en **(IV.33)** et **(IV.34)**.

Définition 12 : (opérateur de restriction flou)

L'opérateur de restriction flou β défini sur les ensembles $\zeta(D_j)$ et $\tilde{A_j}$ correspond à l'ensemble de toutes les paires <p, n> tel que :

$$\zeta(D_j) \text{ β } \tilde{A_j} = \{\langle p,n\rangle \;/\; \exists\, A(X) \in D_j : p = \mu_{\Pi F}(A(X)) \wedge \ldots . n = \mu_{NF}(A(X))\} \qquad \textbf{(IV.32)}$$

où les fonctions d'appartenance ΠF (resp. NF) sont données par :

$$\mu_{\Pi F}(A(X)) = \text{Sup}_{d\in\Delta}\, \text{Min}(\mu_F(d), \pi_{A(X)}(d)) \qquad \textbf{(IV.33)}$$

$$\mu_{NF}(A(X)) = \text{Inf}_{d\in\Delta}\, \text{Max}(\mu_F(d), 1-\pi_{A(X)}(d)) \qquad \textbf{(IV.34)}$$

De ce fait, une base de données possibiliste généralisée a deux interprétations, une interprétation possible résultant de (**IV.33**) et une interprétation certaine résultant de (**IV.34**).

a) Interprétation certaine (nécessaire) d'une base de données possibiliste

Etant donné une relation possibiliste généralisée $\aleph$ définie sur le schéma $\tilde{R}(\tilde{A}_{1,1},\ldots,\tilde{A}_{1,h1},\ldots,\tilde{A}_{j,1},\ldots,\tilde{A}_{j,hj},\ldots,\tilde{A}_{n,1},\ldots,\tilde{A}_{n,hn})$, une interprétation certaine (nécessaire) de $\aleph$, notée N$\aleph$, est définie sur les ensembles flous N$\tilde{A}_{i,j}$ comme suit :

$$\mathbf{N}\aleph \subseteq \mathbf{N}\tilde{A}_{1,1}\times\ldots\times\mathbf{N}\tilde{A}_{1,h1}\times\ldots\times\mathbf{N}\tilde{A}_{j,}\times\ldots\times\mathbf{N}\tilde{A}_{j,hj}\times\ldots\times\mathbf{N}\tilde{A}_{n,1}\times\ldots\times\mathbf{N}\tilde{A}_{n,hn} \quad \textbf{(IV.35)}$$

- Sur le schéma $\tilde{R}$, un tuple (noté $\tilde{t}_i$, i=1..n) de l'instance $\mathbf{N}\aleph$ contient les valeurs certaines (notées $\tilde{t}_i[\tilde{A}_{j,k}]$) de la partition floues correspondante :

$$\tilde{t}_i[\tilde{A}_{j,k}] = \begin{cases} \mathrm{Inf}_{d \in Dj} \ \max(\mu_{Ajk}(d), 1-\pi_{ij}(d)) & \text{si } t_i.A_j = \pi_{ij} \\ \mu_{Ajk}(v_{ij}) & \text{si } t_i.A_j = v_{ij} \quad \text{(valeurs précise)} \end{cases}$$

b) Interprétation possible d'une base de données possibiliste

Etant donné une relation possibiliste généralisée $\aleph$ définie sur le schéma $\tilde{R}(\tilde{A}_{1,1},\ldots,\tilde{A}_{1,h1},\ldots,\tilde{A}_{j,1},\ldots,\tilde{A}_{j,hj},\ldots,\tilde{A}_{n,1},\ldots,\tilde{A}_{n,hn})$, une interprétation possible de $\aleph$, notée $\Pi\aleph$, est définie sur les ensembles flous N$\tilde{A}_{i,j}$ comme suit :

$$\Pi\aleph \subseteq \Pi\tilde{A}_{1,1}\times\ldots\times\Pi\tilde{A}_{1,h1}\times\ldots\times\Pi\tilde{A}_{j,}\times\ldots\times\Pi\tilde{A}_{j,hj}\times\ldots\times\Pi\tilde{A}_{n,1}\times\ldots\times\Pi\tilde{A}_{n,hn} \quad \textbf{(IV.36)}$$

- Sur le schéma $\tilde{R}$, un tuple (noté $\tilde{t}_i$, i=1..n) de l'instance $\Pi\aleph$ contient les valeurs possibles (notées $\tilde{t}_i[\tilde{A}_{j,k}]$) de la partition floues correspondante :

$$\tilde{t}_i[\tilde{A}_{j,k}] = \begin{cases} \mathrm{Sup}_{d \in Dj} \ \mathrm{Min}(\mu_{Ajk}(d), 1-\pi_{ij}(d)) & \text{si } t_i.A_j = \pi_{ij} \\ \mu_{Ajk}(v_{ij}) & \text{si } t_i.A_j = v_{ij} \quad \text{(valeurs précise)} \end{cases}$$

Exemple 6 :

Soit le schéma de la relation *Etudiant*(Nom, Age, Mathematique, Information) donné dans l'exemple du modèle de Prade et Testemale (exemple 5). Le GPRM permet de modéliser les

attributs Age, Mathematique et Informatique par différents ensembles flous. Le schéma de relation correspondant est le suivant :

Ẽtudiant(Nom, Age.Enfant, Age.T_Jeune, Age.Jeune, Age.Adulte, Age.Vieux, Mathématique.Mauvais, Mathématique.Moyen, Mathématique.bon, Informatique.Mauvais, Informatique.Moyen, Informatique.Bon).

Une interprétation en terme de certitude **N**ℵ correspondant à l'extension de la relation possibiliste *Etudiant* est représentée par la table ci-dessous. Une partition floue des attributs Mathematique et Informatique est donnée par la figure **Fig. IV.7**, et une partition floue de l'attribut Age est représentée par la figure **Fig. IV.8**.

	Age					**Mathématiques**			**Informatique**		
Nom	Enfant	T_jeune	Jeune	adulte	vieux	Mauvais	Moyen	bon	Mauvais	Moyen	Bon
Amel	0	0	0.5	0	0	0.4	0.6	0	0	0.6	0
Hider	0	0	1	0	0	0.6	0.4	0	0	0	0
Med Ali	0	0	1	0	0	0.5	0.25	0	0.12	0.5	0
Ramzi	0	0.3	0.55	0	0	0.4	0.6	0	0.12	0.95	0
Kamel	1	1	1	1	1	0	0	0.8	0	0	0.8
Nourdine	0	0.5	0	0	0	0.25	0.8	0	0.95	0	0
Sami	0	0	1	0	0	1	0	0	0.6	0.4	0

Tab. IV.5. Interprétation certaine de la relation *Ẽtudiant.*

	Age					**Mathématiques**			**Informatique**		
Nom	Enfant	T_jeune	Jeune	adulte	vieux	Mauvais	Moyen	bon	Mauvais	Moyen	Bon
Amel	0	0.5	1	0.5	0	0.4	0.6	0	0.4	1	0
Hider	0	0	1	0	0	0.6	0.4	0	0	0	0
Med Ali	0	0	1	0	0	0.8	0.5	0	0.5	0.8	0
Ramzi	0	0.3	0.55	0	0	0.4	0.6	0	0.12	0.95	0
Kamel	1	1	1	1	1	0	0.3	1	0	0.3	1
Nourdine	0.5	1	0.3	0	0	0.25	0.8	0	1	0.12	0
Sami	0	0	1	0	0	1	0	0	0.6	0.4	0

Tab. IV.6. Interprétation possible de la relation *Ẽtudiant.*

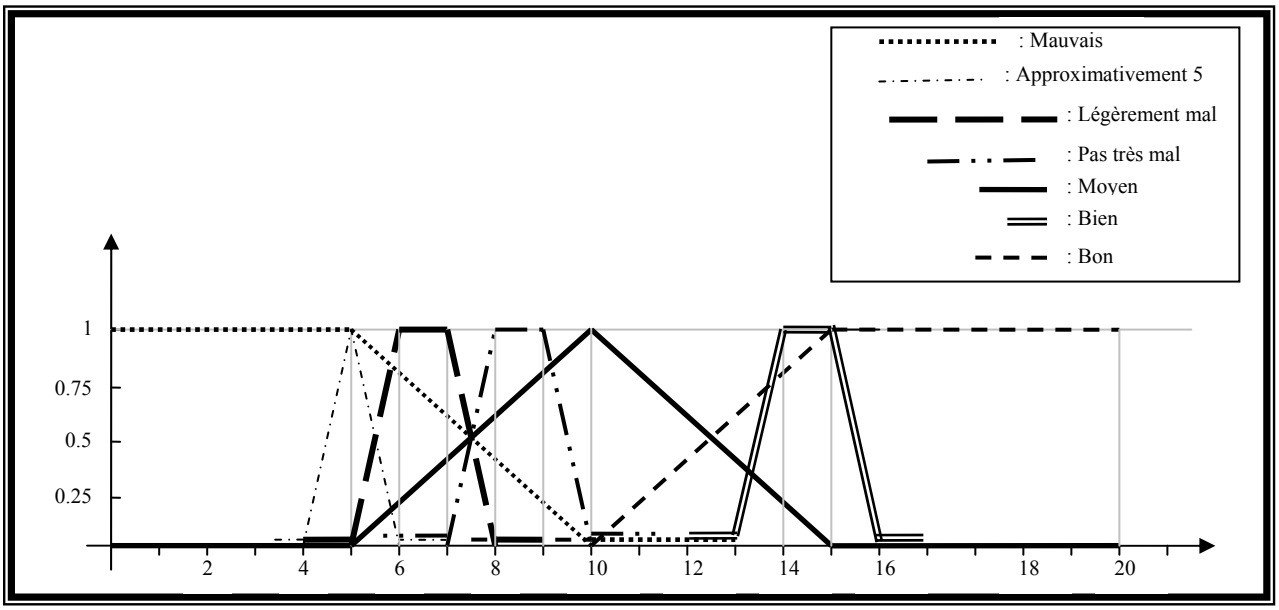


Fig. IV.6. Partition floue des domaines NoteMathématique et NoteInformatique.

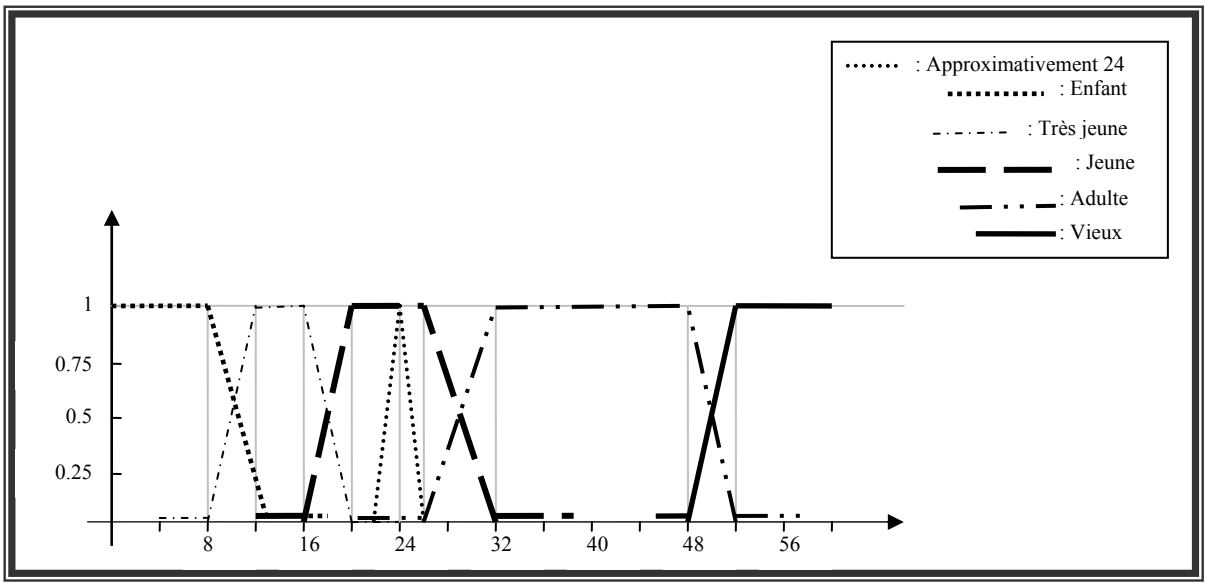


Fig. IV.7. Partition floue du domaine de l'attribut Age.

IV.4- Les Dépendances Fonctionnelles Floues (DFFs)

Par leur utilité incontestable dans l'organisation logique des bases de données, les dépendances fonctionnelles ont aussi été étendues pour supporter les différentes situations où les bases de données sont floues (imprécises).

IV.4.1- Principes des dépendances fonctionnelles floues

Etant donné un schéma de relation ***R***, une instance *r* et deux ensembles d'attributs $X, Y \neq \varnothing$, $X \cap Y = \varnothing$. Une dépendance fonctionnelle floue, notée $X \sim> Y$, peut être appréhendée selon différents principes que nous présentons ci-après :

- Principe de relation floue,
- Principe des valeurs imprécises,
- Principe d'égalité floue,
- Principe de quantificateurs flous.

A)- Le principe de relation floue

Soit *R* une relation floue avec des tuples munis d'un degré d'appartenance du tuple *t* à la relation *r*. Ce type de relation exprime un concept graduel tel que chaque tuple appartient plus ou moins à une classe d'éléments sous-jacente. Le degré d'appartenance du tuple peut alors jouer un rôle dans la satisfaction de la DFF. Dans ce cas, la valeur de vérité de la DFF $X \sim> Y$, notée par $T_R(X \sim> Y)$ est donnée par :

$$T_R(X \rightarrow Y) = \mathrm{Inf}_{t1,\, t2}[\wedge(\mu_R(t_1), \mu_R(t_2), EQ(t_1.X, t_2.X)) \Rightarrow_f EQ(t_1.Y, t_2.Y)] \qquad \textbf{(IV.35)}$$

où $\wedge$ est la conjonction, $\Rightarrow_f$ est une implication floue et $EQ(\boldsymbol{t_1.X}, \boldsymbol{t_2.X}) = 1$ si $\boldsymbol{t_1.X} = \boldsymbol{t_2.X}$, 0 sinon.

Le résultat de cette expression appartient à l'intervalle unité [0, 1] et ne peut être ni la base du théorème de décomposition, ni un outil valide de conception de base de données **[BOS 97b]**.

B)- Le principe de valeurs imprécises

Le second changement de l'expression **(II.2)** apparaît lorsqu'on admet que les valeurs prises par *X* et/ou *Y* sont « imprécisément » connues et représentées par des distributions de

possibilité. Le problème qui se pose alors est au niveau de la manipulation de l'imprécision/incertitude des données présentes dans la base de données.

L'idée est d'étendre la notion de DF sur des valeurs imprécises des attributs. Le point clé (et ainsi la sémantique de la DF étendue) est lié à la comparaison (égalité) entre les valeurs imprécises. En fonction des propositions, la comparaison est basée soit sur les représentations des valeurs, soit sur leur signification dans le cadre possibiliste. Dans les deux cas, le résultat est dans [0, 1] et une implication à valeurs multiples est nécessaire.

C)- Le principe d'égalité floue

La troisième modification de la formule **(II.2)** est la relaxation de l'égalité stricte à une relation de ressemblance (*EQ*). L'idée est de prendre en considération des valeurs de X et Y qui ne sont pas strictement égales mais seulement ressemblantes. Si l'opérateur de ressemblance et l'implication sont choisis convenablement, La DFF $X \sim> Y$ peut entraîner la validité de la DF usuelle $X \rightarrow Y$. l'interprétation canonique de cette extension est : « les plus proches valeurs de X impliquent les plus proches valeurs de Y. Pour l'instant, dans une relation *EMPLOYE*, on peut avoir la propriété : « les plus proches expériences et jobs, les plus proches salaires », en d'autres termes « les employés avec l'expérience et un travail similaires doivent avoir des salaires similaires ». $X \sim> Y$ peut être défini par :

$$\forall\, t_1, t_2 \in r : EQ(t_1.X, t_2.X) \Rightarrow_{RG} EQ(t_1.Y, t_2.Y) \qquad \textbf{(IV.36)}$$

$\Rightarrow_{RG}$: implication de Rescher-Gaines.

La DF classique reste valide et la relation $R(X, Y, Z)$ avec la DFF $X \sim> Y$ peut être évidemment décomposée en ces deux projections $R(X, Y)$ et $R(X, Z)$, comme dans la cas usuel.

D)- Le principe de quantificateur flou

Le changement suivant concerne la relaxation du quantificateur universel (le $\forall\, (t_1, t_2)$) en des quantificateurs flous. Autrement dit, la DF $X \rightarrow Y$ n'est pas vérifiée pour tout couple de tuples $(t_1, t_2) \in r$, il existe des couples tel que $t_1.X = t_2.X$ et $t_1.Y \neq t_2.Y$.

Exemple 7 :

Soit l'extension suivante :

A	*B*	*C*
a_1	b_1	c_1
a_1	b_1	c_2
a_1	b_1	c_3
a_1	b_2	c_4
a_1	b_3	c_5

Pour la DF candidate $A \rightarrow B$ par exemple, les deux dernières valeurs de l'attribut *B* rendent cette DF non valide. Dans ce cas on peut dire que la DF $A \rightarrow B$ est valide à 3/5 = 0.6, i, e. à 60%.

D'un point de vu pratique, ce type d'extension ne peut être utilisée dans le but de concevoir, mais peut fournir quelques méta-connaissances sur l'état courant de la base de données.

Diverses définitions de la notion de dépendance fonctionnelle floue (DFF) ont été proposées depuis 1988. Chacune d'elles traite un ou plusieurs aspects (principe) de l'imprécision de l'information et peut s'intégrer dans un ou plusieurs modèles de base de données relationnelles floues présentés précédemment.

IV.4.2- Les modèles de Dépendances Fonctionnelles Floues

A partir des différents principes énoncés précédemment, différents modèles de dépendances fonctionnelles floues ont été construits et proposés dans la littérature. Nous présentons dans cette section, les plus importants d'entre eux.

A)- Approche de Kiss et al

A. Kiss dans **[KIS 91]** a introduit une définition de DFFs basée sur le principe de relation floue, dite de « type 1 ». Ce type de relations correspond à des relations classiques dont les tuples sont pondérés par leur degré d'appartenance à la relation).

La DFF ($X \sim> Y$) correspond à la formule (II.2), mais sa valeur de vérité n'est plus {0, 1} mais plutôt une valeur $T_R (X \sim> Y) \in [0, 1]$ définie par :

$$\mathbf{T_R(X \sim> Y) = \min_{t1, t2} [\min [\mu_r(t_1), \mu_r(t_2), t_1.X = t_2.X] \Rightarrow_L t_1.Y = t_1.Y]} \quad \mathbf{(IV.37)}$$

Où $\Rightarrow_L$ est l'implication de Lukasiewicz, et $\mu_R(t_1)$ est le degré d'appartenance de t_1 à r.

Cette expression peut être interprétée comme suit : La paire de tuples qui a le plus grand degré d'appartenance, qui viole la DF classique correspond à la DFF ($X \sim> Y$) la moins satisfaite. Le but de cette proposition n'est pas la réduction (élimination) de la redondance, mais plus la minimisation de l'espace de stockage (la réduction de redondance implique la minimisation de l'espace de stockage, alors que la minimisation de l'espace de stockage n'implique pas la réduction de redondance).

L'auteur introduit, principalement, un théorème de décomposition « *léger* ». En effet, si $T_R(X \sim> Y) \geq \theta$, alors le sous-ensemble $r_\alpha(X, Y, Z)$ de $r(X, Y, Z)$ dont le degré d'appartenance de ses tuple est supérieur à α , peut être décomposée à ces deux projections $r_\alpha[X, Y]$ et $r_\alpha[X, Z]$ où $\alpha > 1- \theta$.

B)- Approche de Raju et Mazumder

Une dépendance fonctionnelle floue pour Raju et Mazumdar dans **[RAJ 86][RAJ 88]** est une expression de la forme :

X ~> Y est une DFF valide dans une relation floue *r* sur *R* si et seulement si :

$$\forall\, t_1, t_2 \in r \quad EQ(t_1.X, t_2.X) \Rightarrow_{RG} EQ(t_1.Y, t_2.Y) \qquad \textbf{(IV.38)}$$

Où $\Rightarrow_{RG}$ est l'opérateur d'implication de Rescher-Gaines, et *EQ* est une relation de ressemblance, définie par :

$$EQ_X(t_1, t_2) = \mathbf{Min}_u\ 1 - | \pi_{t1.X}(u) - \pi_{t2.X}(u) | \qquad \textbf{(IV.39)}$$

Cette notion de DFF peut être appliquée aux bases de données classiques comme elle peut être appliquée aux relations floues contenant des tuples pondérés, elle est plus forte que la notion de DF dans les relations classiques.

C)- Approche de Prade et Tastemale

Prade et Testemale **[PRA 84]** ont considéré le modèle possibiliste de données floues avec des contraintes de DFFs tel que « l'âge détermine approximativement le salaire », où la relation peut avoir quelques valeurs d'attributs imprécis, et sont décrits au moyen de

distributions de possibilité. Les auteurs ont présenté la définition suivante : Une DFF $X \sim> Y$ est valide dans r, si et seulement si :

$$\forall t_1, t_2 \in r : \; [t_1.X = t_2.X] \Rightarrow [EQ(t_1.Y, t_2.Y)] \geq \lambda \qquad \textbf{(IV.40)}$$

Où λ est un seuil fixé et EQ une relation de ressemblance utilisée pour comparer une paire des valeurs classiques ou leurs de distributions possibilité. Quand EQ est appliquée aux distributions de possibilité, elle calcule le degré d'égalité des valeurs imprécises $t_1.X$ et $t_2.X$ en utilisant la formule **(IV.41)** ci-dessous ($t_1.X$ et $t_2.X$ sont représentés respectivement par les distributions de possibilité $\pi_{t1.X}$ et $\pi_{t2.X}$). EQ est une fonction d'un sous-ensemble de **dom**X $\rightarrow$ **[0, 1]** définie par :

$$EQ(t_1.X, t_2.X) = \mathbf{Sup}_{x \in \mathbf{dom}\, X} \; \mathbf{Min}(\pi_{t1.X}(x), \pi_{t2.X}(x)) \qquad \textbf{(IV.41)}$$

D)- Approche de Bhuniya et Niyogi

Bhuniya et Niyogi **[BHU 93]** ont donné une définition de DFF en considérant les contraintes de type : « les salaires des employés ayant des qualifications presque égales devrait être plus ou moins égal ». Par cette approche, une DFF $X \sim> Y$ est valide dans r si et seulement si :

$$\forall t_1, t_2 \in r : EQ(t_1.X, t_2.X) \Rightarrow_{RG} EQ(t_1.Y, t_2.Y)$$
$$\text{ou } EQ(t_1.X, t_2.X) - EQ(t_1.Y, t_2.Y) \leq 1-\beta \qquad \textbf{(IV.42)}$$

où $EQ(t_1.X, t_2.X) \geq \alpha$ et $EQ(t_1.Y, t_2.Y) \geq \alpha$, et $\alpha < \beta < 1$

Les résultats suivants ont été établis :

- Une axiomatique étendue saine et complète,
- Le théorème de décomposition sans perte d'information est valide et un algorithme pour tester la préservation de DFFs quand la décomposition est appliquée.

E)- Approche de Cubero et al

Cubero et al. **[CUB 93][CUB 94]** considèrent qu'une DFF $X \sim>_{(\alpha, \beta)} Y$ est valide dans r si et seulement si :

$$\forall t_1, t_2 \in r : EQ(t_1.X, t_2.X) \geq \alpha \Rightarrow EQ(t_1.Y, t_2.Y) \geq \beta \qquad \textbf{(IV.43)}$$

Cela signifie que si $t_1.X$ est X-ressemblant à $t_2.X$ à un degré au minimum α, alors $t_1.Y$ doit être Y-ressemblant à $t_2.Y$ à un degré au minimum β. Cette définition ne capture pas la sémantique de la relation fonctionnelle. Si $X \sim>_{(\alpha, \beta)} Y$ est valide, $X \rightarrow Y$ n'est pas nécessairement valide dans cette relation. L'inverse aussi n'est pas vrai (si $X \rightarrow Y$ est valide cela n'implique pas que $X \sim>_{(\alpha, \beta)} Y$ est valide). Puisque dans une relation qui satisfait la DF $X \rightarrow Y$, il peut avoir deux tuples satisfaisant $EQ(t_1.X, t_2.X) \geq \alpha$ et $EQ_Y(t_1.Y, t_2.Y) < \beta$, ce qui viole $X \sim>_{(\alpha, \beta)} Y$.

Aussi la DFF $X \sim>_{(1, 1)} Y$ n'implique pas la DF $X \rightarrow Y$ à moins qu'on restreint la relation de ressemblance comme dans la proposition de Raju et Majumder **[RAJ 86][RAJ 88]** à : $EQ(a, b) = 0$ pour $a \neq b$.

F)- Approche de Chen et al

Chen **[CHE 95]** et Chen et al. **[CHE 95b]** ont défini de manière similaire la DFF sur des relations floues. Pour un seuil λ donné, une DFF, notée $\boldsymbol{X \sim>_{\lambda} Y}$ **($\lambda \in$] 0, 1]**), est valide sur r si et seulement si :

$$\forall t_1, t_2 \in r : \text{Si } t_1.X = t_2.X \text{ Alors } t_1.Y = t_2.Y$$
$$\text{Sinon } [EQ(t_1.X, t_2.X) \Rightarrow_G EQ(t_1.Y, t_2.Y)] \geq \lambda \qquad \textbf{(IV.44)}$$

Où $\Rightarrow_G$ est l'opérateur de l'implication de Gôdel.

Cette définition d'une DFF est également plus forte que la notion de DF dans des relations classiques, c'est-à-dire si la DFF $X \sim>_{\lambda} Y$ est valide dans la relation floue r, alors $X \rightarrow Y$ est aussi valide dans r.

Les résultats suivants ont été démontrés :

- Un ensemble d'axiomes d'inférence étendu complet et sain,
- Le théorème de décomposition sans perte d'information est valide,
- Un algorithme pour tester la préservation des DFFs quand la décomposition est appliquée.

Cependant, cette définition présente une certaine discontinuité dans la dépendance fonctionnelle des valeurs de Y sur valeurs de X, puisque si la ressemblance de des valeurs de Y est supérieur ou égal à λ, peut importe si les valeurs de X, associés à ces valeurs de Y,

ne sont pas égales. Ainsi, la sémantique de la définition n'est pas strictement fonctionnelle par nature.

G)- Approche de Bosc et al

Dans les approches précédentes, quelques problèmes ont été mis en évidence, en particulier le respect du sens de la DF en présence de données imprécises. Pour surmonter ces problèmes, Bosc et al. **[BOS 97b]** ont suggéré une approche où aucun seuil n'apparaît et la DF classique est étendue dans le sens que la caractéristique fonctionnelle est préservée dans les deux niveaux de représentation (classique et flou) et ainsi le théorème de décomposition est applicable. Pour cela, ils ont utilisé « les règles graduelles » **[DJO 07b]** et « les règles de certitude ». La définition proposée alors pour une dépendance fonctionnelle floue est la suivante :

$\boldsymbol{X \sim> Y}$ est une DFF valide dans r si et seulement si :

$$\forall\, t_1, t_2 \in r :\ \text{Si } t_1.X = t_2.X \text{ Alors } t_1.Y = t_2.Y$$
$$\text{Sinon } EQ_X(t_1, t_2) \Rightarrow_{RG} EQ_Y(t_1, t_2) \qquad \textbf{(IV.45)}$$

où $\Rightarrow_{RG}$ est l'implication de Rescher-Gaines et la relation EQ est définie par : $EQ(t_1.X, t_2.X)=\text{Sup}_{x\in \text{dom}X}\ \text{Min}\ (\pi_{t1.X}(x), (\pi_{t2.X}(x))$.

Il vaut la peine de noter que la première partie de la définition traite la fonction au niveau de la représentation (qui est purement syntaxique), quand à la deuxième partie elle exprime la contrainte au niveau des valeurs elles même en terme de possibilité d'existence de fonction entre X et Y.

IV.5- Etat de l'art sur la découverte des DFFs

Malgré la multitude de modèles de dépendances fonctionnelles floues, il n'existe dans la littérature (à notre connaissance) qu'un seul travail concernant la découverte de DFFs à partir de bases de données contrairement aux dépendances fonctionnelles classiques qui ont reçu beaucoup d'attention (voir chapitre III).

En effet, sur le problème de l'inférence des DFFs, nous n'avons pu trouver qu'un seul travail qui traite ce problème. Dans ce travail **[WAN 00]**, l'auteur propose un algorithme dit "Level-Wise search of functional dependencies algorithm" ou "algorithme de Wang".

Algorithme de Wang

L'algorithme proposé par Wang à pour objectif la découverte des dépendances fonctionnelles floues définies par la formule **(IV.45)** :

Cet algorithme se base sur l'intégration de la procédure COMPUTE-DEPENDENCIES dans l'algorithme TANE de Huhtala. L'algorithme 1 présente le pseudo-code de cette approche.

Algorithme 1 : level-wise search of functional dependencies [WAN 00]

Entrée : r, une relation sur R
Sortie : F, une couverture de dep(r)
1 : $F \leftarrow \varnothing$
2 : $L_0 \leftarrow \{\varnothing\}$
3 : $C^+(\varnothing) \leftarrow R$
4 : $L_1 \leftarrow \{\{A\} \mid A \in R\}$
5 : $i \leftarrow 1$
6 : **while** $L_i \neq \varnothing$ **do**
7 : **for every** $X \in L_i$, COMPUTE-RHS-CANDIDATES $C^+(X)$
8 : PRUNE (L_i, F)
9 : COMPUTE-DEPENDENCIES(L_i, F)
10 : $L_{i+1} \leftarrow$ GENERATE-NEXT-LEVEL(L_i, F)
11 : $i \leftarrow i + 1$
12 : **end while**

Remarque :

- Les procédures COMPUTE-RHS-CANDIDATES et PRUNE sont les mêmes que dans l'algorithme TANE.

La procédure COMPUTE-DEPEDENCIES

La procédure COMPUTE-DEPENDENCIES dans l'approche de Wang commence par calculer pour chaque paire de tuple $(t_1, t_2) \in r$ et pour chaque partie gauche $X \in r$, $EQ(t_1.X, t_2.X)$ de la manière suivante.

- Si la partie gauche X est réduite à un seul attribut, alors :

$$EQ(t_1.X, t_2.X) = \mathrm{Sup}_{x \in \mathrm{dom}\, X}\ \mathrm{Min}(\pi_{t1.X}(u), (\pi_{t2.X}(u))$$

- Si la partie gauche X est un sous-ensemble d'attributs ($X = X_1X_2\ldots X_n$), alors :

$$EQ(t_1.X, t_2.X) = \mathrm{Max}(EQ(t_1.\boldsymbol{X_1}, t_2.\boldsymbol{X_1}), EQ_{\boldsymbol{X2}}(t_1.\boldsymbol{X_2}, t_2.\boldsymbol{X_2}),\ldots, EQ(t_1.X_n, t_2.X_n))$$

L'algorithme 2 représente la procédure COMPUTE-DEPENDENCIES.

Algorithme 2 : COMPUTE-DEPENDENCIES (L_i, F)

---

1. **For every** attribute A_i in X and A_R, calculate $EQ_A(t_i, t_j)$ between every two tuples t_i and t_j in r,
2. Check $EQ_X(t_i, t_j) \leq EQ_A(t_i, t_j)$ for every i, j in r. **If** true, then the *else* part of the dependency in (IV.45) is satisfied,
3. **If** step (2) is true and $EQ_X(t_i, t_j) = EQ_A(t_i, t_j) = 1$, **then** check $t_1.X = t_2.X \Rightarrow t_1.Y = t_2.Y$. **If** true, then the functional dependency in (IV.45) holds.

Limites de l'approche de Wang

L'approche de Wang présentée ci-dessus n'est pas optimale dans la mesure où :

- Aucun élagage n'est effectué sur l'espace des tuples. De ce fait cet algorithme peut être assimilé à l'algorithme naïf de la découverte des dépendances fonctionnelles (cf. § **III.4.2.1**). La considération de tout l'espace de tuple dans chaque étape de l'algorithme représente une contrainte importante pour des bases de données volumineuses. Sa complexité algorithmique est de l'ordre de $|\mathbf{R}| * 2^{|\mathbf{R}|-1}$.
- L'approche est proposée pour une définition de DFF qui est une règle implicative mais ne respect pas la sémantique des dépendances fonctionnelles classiques.
- De plus, l'utilisation de la co-norme « Max » dans la relation *EQ* sur des sous-attributs (de taille supérieure à 1) semble est inappropriée.

IV.6- Conclusion

La notion de « fuzzification » du modèle relationnel touche presque tous les fondements théoriques du modèle relationnel défini par CODD. Nous citons : les valeurs des champs, le domaine de définition des attributs, la définition d'une dépendance fonctionnelle et même les opérateurs relationnels. Dans ce chapitre, nous avons présenté les principaux modèles flous, sans arborer la « fuzzification » de l'algèbre relationnelle. Quand aux définitions des DFFs vues dans ce chapitre, elles apportent toutes des visions différentes (selon la signification donnée au terme « Flou »).

Dans le prochain chapitre, nous allons apporter à notre tour une nouvelle définition aux dépendances fonctionnelles floue, qui sera adaptée à notre conception de la « fuzzification », tout en proposant la partie axiomatique et algorithmique permettant de découvrir ce type de DFFs à partir d'une base de données existante, par des techniques de Data Mining.

CHAPITRE 5

CONTRIBUTIONS

Contrairement à la notion de dépendance fonctionnelle qui est définie de manière unique dans le cas de base de données classiques, la notion de dépendance fonctionnelle floue a plusieurs définitions présentes dans la littérature (**cf. §IV.4**). Cette diversité des DFFs provient :

D'une part, de la multitude de raisons pouvant rendre une base de données floue. Ces raisons ne sont pas exclusives mutuellement et peuvent être résumés en trois points essentiels:

- Bases de données avec domaines d'attributs flous : ce qui signifie que les valeurs des données sont imprécises et peuvent être représentées par : des ensembles flous, des distributions de possibilité, des variables linguistiques ou bien même des valeurs manquantes),
- Bases de données avec des tuples flous : ce qui signifie que les valeurs des attributs peuvent être *précises* mais le tuple lui-même est pondéré par un degré d'appartenance à la base de données,
- Bases de données avec des schémas de relation flous : ce qui signifie que les attributs peuvent être des partitions floues, comme c'est le cas avec le modèle GPRM.

D'autre part, l'interprétation de cette « fuzzification », sur une même base de données peut être différente d'un auteur à un autre. Autrement dit, la sémantique des DFFs peut être différente. Par exemple, les définitions de **Cubero et al.** et de **Chen et al.** ont des sémantiques différentes malgré qu'elles sont destinées au même type de bases de données floues (base de données possibilistes).

En dépit de l'intérêt qu'ont reçu les DFFs, il existe peu de travaux traitant leur inférence à partir de bases de données existantes. Dans ce chapitre, nous proposons une approche complète d'Extraction de Connaissance à partir de Données (ECD) dédiée à la découverte des DFFs à partir de bases de données possibilistes. Cette approche basée sur le concept de *stratification* sera détaillée dans la section **§V.2**. Mais avant, nous expliquons d'abord la préparation et la mise en forme des données cibles qui est une phase de l'étape de consolidation dans tout processus d'ECD.

V.1- Préparation des données cibles

Comme nous l'avons vu dans le chapitre précédent (cf. **§IV.3**), il existe une multitude de modèles de données floues et un nombre assez conséquent de définitions de DFFs. Par conséquent, il est difficile de proposer une approche globale pour l'inférence des DFFs sans définir les modèles de données floues à traiter et la définition de DFF adoptée. La sous-section suivante présente de manière très synthétique les différentes formes que peuvent avoir les données dans les bases de données imprécises.

V.1.1- Choix du modèle cible et fuzzification des données

Les bases de données floues auxquelles nous nous intéressons dans le cadre de notre travail sont des bases de données dont l'imprécision touche essentiellement les valeurs de données. Par conséquent, les modèles relationnels flous présents dans la littérature prenant en charge cette catégorie de bases de données sont : le modèle d'Umano-Fukami, le modèle de Prade et Testemale, le modèle GEFRED et le modèle GPRM. Les valeurs des données dans ces modèles peuvent être sous plusieurs formes possibles. Ces formes sont récapitulées par le tableau suivant.

Type de données	Notation	Exemple
Valeurs précises	v_{ij}	t_1.Age = 45
Distribution de possibilité	π_{ij}	t_1.Salaire = 0.5/30+1/38+0.8/40
Ensemble flou	μ_{ij}	t_3.Age = 0.8/27
Variables linguistiques	L_{ij}	t_4.Salaire = *bas*
Valeurs manquantes	Missing Value	t_2.Job = *Missing value*

Tab.V.1. Récapitulatif des différentes formes de données traitées

L'exemple suivant illustre une relation comportant ce type de valeurs.

Exemple 1 :

Soit l'instance de relation *EMPLOYE*(Nomemp, Age, Job, Salaire) représentée par le table suivante :

#	**Nomemp**	**Age** (année)	**Job**	**Salaire** (mille DA)
1	Aissat	45	Chef de service	0.5/30+1/38+0.8/40
2	Foudhil	31	*Missing value*	27
3	Kaidi	0.8/27	Ingénieur	23.6
4	Ait said	*Jeune*	Chauffeur	*bas*

Dans notre travail, nous proposons d'effectuer un "mapping" de ces modèles vers un modèle purement possibiliste (fuzzification des données). Et cela a pour but d'unifier la représentation des données (sous forme de distributions de possibilité) afin de faciliter la comparaison de leurs valeurs par une relation de ressemblance unique.

Les procédures de fuzzification des différentes représentations de données traitées, afin qu'elles soient directement exploitables par l'algorithme proposé pour l'inférence des DFFs, sont résumées dans le tableau suivant :

Types de données floues	**Fuzzification**	**Exemple (Age)**
Valeurs précises	Attribuer un degré de possibilité égal à 1 pour la valeur de $t_i.A_j$	t_1.Age = 1.0/45
Distribution de possibilité	Aucune fuzzification	t_1.Salaire = 0.5/30+1/38+0.8/40
Ensemble flou	Aucune fuzzification	t_3.Age = 0.8/27
Variables linguistiques	Remplacer par sa distribution de possibilité	t_4.Salaire = 0.6/10+1.0/13+ 0.4/15
Valeurs manquantes	Appliquer la transformation de Dubois et Prade	*Cette méthode est détaillée dans la section* V.2.

Tab.V.2. Fuzzification des données.

Remarque très importante

La présence de valeurs manquantes est un phénomène indésirable dans une base de donnée, car elle cause de nombreux problèmes lors des différents traitements effectués sur ces bases. Prenant le cas de dépendances fonctionnelles, l'inférence de ces dernières à partir de bases de données existantes impose la comparaison de toutes les valeurs des

tuples pour chaque attribut. Dans le cas où des valeurs manquantes existent dans la base, cette comparaison devient impossible. Pour cela, il est nécessaire de générées des valeurs substitutives pour remplacer les valeurs manquantes. La transformation de Dubois et Prade est une méthode pouvant être utilisée pour palier au problème des valeurs manquantes.

V.1.2- Traitement des valeurs manquantes

L'application de la transformation de Dubois et Prade, lors de la fuzzification des données (sur les valeurs manquantes), permet de générer des distributions de possibilité pour ces valeurs en utilisant les distributions de probabilités sous-jacentes. Ce qui fait que dans notre cas (base de données), pour tout attribut contenant des valeurs manquantes on génère d'abord la distribution de probabilités des données représentée par la *fréquence d'apparition* des différentes valeurs du domaine d'un attribut dans la relation considérée.

Le problème de transformation des probabilités en possibilités a suscité de nombreux travaux **[DUB 86][CIV 86][DEL 87][LAS 00][DUB 00b]**. Un principe de consistance entre probabilités et possibilité a été pour la première fois posé par Zadeh **[ZAD 78]** de façon informelle : *ce qui est probable devrait être possible*. Dubois et Prade ont ensuite traduit ce principe par l'inégalité :

$$\mathbf{P}(A) \leq \mathbf{\Pi}(A) \quad \forall A \subseteq \Omega \tag{V.1}$$

où $\mathbf{P}$ et $\mathbf{\Pi}$ désigne, respectivement, une mesure de probabilité et de possibilité sur un domaine $\Omega = \{\omega_1, ..., \omega_K\}$. Dans ce cas, on dit que $\mathbf{\Pi}$ *domine* $\mathbf{P}$. Transformer une mesure de probabilité en une mesure de possibilité revient à choisir une mesure de possibilité dans l'ensemble $\mathcal{F}(\mathbf{P})$ des mesures de possibilité dominant $\mathbf{P}$. Dubois et al. **[DUB 91][DUB 04]** ont proposé d'ajouter les contraintes suivantes, qui assurent la préservation de la forme de la distribution (contraintes de préservation d'ordre strict) :

$$\mathbf{p}_i < \mathbf{p}_j \Leftrightarrow \pi_i < \pi_j \quad \forall\, i, j \in \{1, \ldots, K\} \tag{V.2}$$

où $\mathbf{p}_i = \mathbf{P}(\{\omega_i\})$ et $\pi_i = \mathbf{\Pi}(\{\omega_i\})$, pour tout $\boldsymbol{i \in \{1, \ldots, K\}}$. Il alors naturel de chercher la distribution de possibilité $\boldsymbol{\pi}$ *la plus spécifique* vérifiant (V.1) et (V.2). On rappelle qu'une distribution de possibilité $\boldsymbol{\pi}$ est plus spécifique que π' si $\pi_i \leq \pi_i'$, $\forall\, i$. Dubois et Prade **[DUB 91][DUB 04]** montrent que la solution à ce problème existe et est unique. Elle peut

être décrite de la façon suivante. En supposant que $p_i \neq p_j$ quel que soit i, j $(i \neq j)$, il est possible de définir une relation d'ordre stricte $\mathcal{L}$ sur $\Omega = \{\omega_1, ..., \omega_K\}$ telle que :

$$(\omega_i, \omega_j) \in \mathcal{L} \Leftrightarrow p_i < p_j. \quad \textbf{(V.3)}$$

Soit σ une permutation des indices $\{1, \ldots, K\}$ associée à cet ordre strict telle que $p_{\sigma(1)} < p_{\sigma(2)} < \ldots < p_{\sigma(K)}$ ou, de façon équivalente :

$$\sigma(i) < \sigma(j) \Leftrightarrow (\omega_{\sigma(i)}, \omega_{\sigma(j)}) \in \mathcal{L}. \quad \textbf{(V.4)}$$

La permutation σ est une bijection et la transformation inverse σ^{-1} donne le rang de chaque p_i dans la liste des probabilités triées par ordre croissant. On peut alors exprimer la transformation de Dubois et Prade sous la forme suivante :

$$\forall i. \quad \pi_i = \sum_{\{j \mid \sigma^{-1}(j) \leq \sigma^{-1}(i)\}} p_j \quad \textbf{(V.5)}$$

Exemple 2 :

Soit $p_1 = 0.2$, $p_2 = 0.35$, $p_3 = 0.4$, et $p_4 = 0.05$. Alors on a $\sigma(1) = 4$, $\sigma(2) = 1$, $\sigma(3) = 2$, $\sigma(4) = 3$ and $\sigma^{-1}(1) = 2$, $\sigma^{-1}(2) = 3$, $\sigma^{-1}(3) = 4$, $\sigma^{-1}(4) = 1$. La transformation **(V.5)** donne donc la distribution de possibilité suivante :

$\pi_1 = p_1 + p_4 = 0.2 + 0.05 = 0.25$

$\pi_2 = p_2 + p_1 + p_4 = 0.35 + 0.2 + 0.05 = 0.6$

$\pi_3 = p_3 + p_2 + p_1 + p_4 = 0.4 + 0.35 + 0.2 + 0.05 = 1$

$\pi_4 = p_4 = 0.05$.

Remarque :

La formulation **(V.5)** suppose que les valeurs de p_i soient toutes différentes. Si au moins deux valeurs sont égales, **(V.3)** n'induit pas un ordre strict, mais un ordre partiel $\mathcal{P}$ sur Ω. Cet ordre partiel peut être représenté par l'ensemble de ses extensions linéaires $\Lambda(\mathcal{P}) = \{\mathcal{L}_l, l = 1, L\}$, telle que chaque extension linéaire est obtenue en choisissant un ordre sur les valeurs égales. A chaque extension linéaire possible $\mathcal{L}_l$ de $\Lambda(\mathcal{P})$, correspond une permutation σ_l de l'ensemble $\{1, ..., K\}$ telle que :

$$\sigma_l(i) < \sigma_l(j) \Leftrightarrow (\omega_{\sigma l(i)}, \omega_{\sigma l(j)}) \in \mathcal{L}_l. \quad \textbf{(V.6)}$$

Dans ce cas, la distribution la plus spécifique compatible avec $p = (p_1, ..., p_K)$ est obtenue en retenant le maximum sur toutes les permutations possibles :

$$\forall i, \quad \boldsymbol{\pi_i = \mathrm{Max}_{l=1,L} \sum_{\{j \mid \sigma_l^{-1}(j) \leq \sigma_l^{-1}(i)\}} p_j} \qquad \textbf{(V.7)}$$

Exemple 2 :

Soient $p_1 = 0.2$, $p_2 = 0.5$, $p_3 = 0.2$, et $p_4 = 0.1$. Il y a donc deux permutations possibles $\sigma_1(1) = 4$, $\sigma_1(2) = 1$, $\sigma_1(3) = 3$, $\sigma_1(4) = 2$ et $\sigma_2(1) = 4$, $\sigma_2(2) = 3$, $\sigma_2(3) = 1$, $\sigma_2(4) = 2$.

En appliquant la transformation (V.7), on obtient :

$\pi_1 = \mathrm{Max}(p_4 + p_1, p_4 + p_3 + p_1) = \mathrm{Max}(0.3, 0.5) = 0.5$
$\pi_2 = p_4 + p_1 + p_3 + p_2 = 1$
$\pi_3 = \mathrm{Max}(p_4 + p_1 + p_3, p_4 + p_3) = \mathrm{Max}(0.5, 0.3) = 0.5$
$\pi_4 = p_4 = 0.1$.

Remarque : On voit que $p_1 = p_3$ implique que $\pi_1 = \pi_3$, une condition imposée par la préservation des ordres stricts.

V.2- Présentation de l'approche stratifiée

L'approche stratifiée est dédiée aux bases de données floues comportant les différents types de données floues présentées dans le tableau **Tab.V.1**. Le résultat des différentes transformations (fuzzifications) est une base de données dont les valeurs des attributs sont représentées par des distributions de possibilité. Cette approche se base sur la proposition d'une nouvelle définition de DFF donnée dans la sous-section §**V.2.1**. En suite, un ensemble de règles d'inférence de ce nouveau type de DFFs seront prouvés dans la sous-section §**V.2.2**. Enfin, le processus complet de découverte de ces DFFs, en s'inspirant des algorithmes de Data Mining pour l'inférence des DFs, sera détaillé dans la sous-section §**V.2.3**.

V.2.1- Proposition d'une nouvelle définition de dépendance fonctionnelle floue

Nous avons choisi de proposer une nouvelle définition de DFF sur laquelle notre approche sera basée. Cette définition s'inspire du fait que la condition d'égalité des tuples dans le cas des DFs doit être traduite par une relation de ressemblance forte entre les couples de tuples dans le cas de DFFs. Pour cela nous utilisons la notion de coupe de niveau α sur l'ensemble des couples de tuples pour ne sélectionner que ceux dont le degré de ressemblance est supérieur ou égal à un seuil α fixé. Notre définition de DFF prend en considération deux principes de fuzzification (cf. **IV.4.1**), qui sont :

(B) le principe des valeurs imprécises (les valeurs sont des distributions de possibilité),

(C) le principe de l'égalité floue (la relation d'égalité entre les valeurs est une relation de ressemblance).

Le choix de ces deux principes n'est pas arbitraire, il a été plutôt imposé par le type de bases de données floues que nous manipulons. (i,e. Base de données où l'imprécision est modélisée par des distributions de possibilité). La définition formelle de DFF est la suivante :

Définition 1 : (dépendance fonctionnelle floue de niveau α)

Une dépendance fonctionnelle floue de niveau α (en abrégé : α-DFF), notée $\boldsymbol{X \sim>_{\alpha} Y}$ ($\alpha \in]0, 1]$), avec $X, Y \subset \boldsymbol{R}$, est valide dans une relation floue (possibiliste) $\tilde{r}$ sur $\boldsymbol{R}$ si, et seulement si :

$$\forall\, t_1, t_2 \in \tilde{r} : \text{Si } \boldsymbol{t_1.X = t_2.X} \text{ Alors } \boldsymbol{t_1.Y = t_2.Y}$$
$$\text{Sinon } \boldsymbol{EQ_X(t_1, t_2) \geq \alpha \Rightarrow EQ_Y(t_1, t_2) \geq EQ_X(t_1, t_2)} \qquad \textbf{(V.8)}$$

Où EQ_X est une relation de ressemblance définie par :

$$EQ_X(t_1, t_2) = \text{Sup}_{u \in \text{dom}(X)} \text{Min}(\pi_{t1.X}(u), (\pi_{t2.X}(u)). \text{ Si } X \text{ est un seul attribut.} \qquad \textbf{(V.9)}$$

- Si X est un sous-ensemble d'attributs de $\boldsymbol{R}$. ($X = A_1A_2\ldots A_k$), alors on aura :

$$EQ_X(t_1, t_2) = \text{Min}(EQ_{A1}(t_1, t_2), EQ_{A2}(t_1, t_2),\ldots, EQ_{Ak}(t_1\, t_2)). \qquad \textbf{(V.10)}$$

La définition **(V.8)** des α-DFFs peut être formulée d'une autre manière. Le lemme 1 représente une autre manière d'exprimer l'expression **V.8** en utilisant les ensembles flous. Ces ensemble sont définis comme suit :

Soit $\mathcal{X}$ un ensemble flou dont les éléments sont des paires de tuples $(t_i, t_j, X\alpha_{ij}) \in \tilde{\boldsymbol{r}} \times \tilde{\boldsymbol{r}}$, où $X\alpha_{ij}$ est le degré de ressemblance de t_i et t_j. L'ensemble $\mathcal{X}$ est défini par :

$$\mathcal{X} = \{((t_i, t_j), X\alpha_{ij}) \mid (t_i, t_j) \in \tilde{r} \times \tilde{r} : i = 1..(m\text{-}1),\ j = 2..m\} \qquad \textbf{(V.11)}$$

et soit $\mathcal{X}_\alpha$ la coupe de niveau α de l'ensemble $\mathcal{X}$. $\mathcal{X}_\alpha$ est donc définit comme suit :

$$\mathcal{X}_\alpha = \{((t_i, t_j), X\alpha_{ij}) \mid X\alpha_{ij} \geq \alpha\} \qquad \textbf{(V.12)}$$

Lemme 1 :
Une α-DFF $X \sim>_\alpha Y$ peut être exprimée en terme d'inclusion floue de l'ensemble $\mathcal{X}_\alpha$ dans l'ensemble $\mathcal{Y}_\alpha$. La deuxième partie de l'expression **(V.8)** est équivalente à :

$$\mathcal{X}_\alpha \subseteq_f \mathcal{Y}_\alpha \qquad \textbf{(V.13)}$$

Remarque :

- Contrairement aux coupes de niveau α qui sont des ensembles ordinaires (non flous), l'ensemble $\mathcal{X}_\alpha$ garde trace du degré d'appartenance de ces éléments à l'ensemble $\mathcal{X}$.

  $\subseteq_f$ est l'inclusion floue définie par l'expression **(IV.17)**.

- Le lemme 1 sera utilisé essentiellement pour prouver certaines règles d'inférence et pour la formalisation d'une coupe de niveau α dans la partie algorithmique.

V.2.2- Axiomes d'Armstrong

Pour prouver la correction des axiomes d'Armstrong correspondants à la définition 1, considérons une relation $\tilde{r}$ sur $\boldsymbol{R}(A_1,\ldots, A_n)$ et deux tuples quelconques t_1 et t_2 de $\tilde{r}$. Les règles d'inférence des DFs s'étendent dans le cas flou de la manière suivante :

FF1. Réflexivité: Si $\boldsymbol{Y \subseteq X}$ est valide**,** alors $\boldsymbol{X \sim>_{\alpha} Y}$ est valide.

Preuve :

Soit $X = A_1A_2\ldots A_k$, et soit $A_i \subseteq X$ tel que :

$EQ_X(t_1, t_2) = \text{Min}(EQ_{A1}(t_1, t_2),\ldots, EQ_{Ak}(t_1, t_2)) = EQ_{Ai}(t_1, t_2)$

- On veut montrer : $\forall\ t_1, t_2 \in \tilde{r} : EQ_X(t_1, t_2) \geq \alpha \Rightarrow EQ_Y(t_1, t_2) \geq EQ_X(t_1, t_2)$

- Par hypothèse : $Y \subseteq X$. on aura donc deux cas possibles :

a) $A_i \subseteq Y$

$(A_i \subseteq Y) \Rightarrow EQ_Y(t_1, t_2) = EQ_{Ai}(t_1, t_2) \Rightarrow EQ_X(t_1, t_2) = EQ_Y(t_1, t_2)$

D'où : $\boldsymbol{X \sim>_{\alpha} Y}$ est vérifiée.

b) $A_i \not\subset Y$

$A_i \not\subset Y \Rightarrow [EQ_Y(t_1, t_2) \geq EQ_{Ai}(t_1, t_2)] \Rightarrow [EQ_Y(t_1, t_2) \geq EQ_X(t_1, t_2)] \Rightarrow [\forall\ t_1, t_2 \in \tilde{r} :$

$EQ_X(t_1, t_2) \geq \alpha \Rightarrow EQ_Y(t_1, t_2) \geq EQ_X(t_1, t_2)]]$.

D'où, $\boldsymbol{X \sim>_{\alpha} Y}$ est vérifiée.

De **a)** et **b)**, on aura donc : une α-DFF $\boldsymbol{X \sim>_{\alpha} Y}$ est réflexive. □

FF2. Augmentation : Si $\boldsymbol{X \sim>_{\alpha} Y}$ est valide, alors $\boldsymbol{XZ \sim>_{\alpha} YZ}$ est valide.

Preuve :

- Par hypothèse on a : $\boldsymbol{X \sim>_{\alpha} Y}$ est valide dans $\tilde{r}$ cela veut dire :

$\forall\ t_1, t_2 \in \tilde{r}, EQ_X(t_1, t_2) \geq \alpha \Rightarrow EQ_Y(t_1, t_2) \geq EQ_X(t_1, t_2)$

Soit Z un sous-ensemble d'attributs de $\boldsymbol{R}$. on aura donc :

$EQ_{XZ}(t_1, t_2) = \text{Min}(EQ_X(t_1, t_2), EQ_Z(t_1, t_2))$ et $EQ_{YZ}(t_1, t_2) = \text{Min}(EQ_Y(t_1, t_2), EQ_Z(t_1, t_2))$

- On veut montrer que : $\forall\ t_1, t_2 \in \tilde{r}, EQ_{XZ}(t_1, t_2) \geq \alpha \Rightarrow EQ_{YZ}(t_1, t_2) \geq EQ_{XZ}(t_1, t_2)$.

- Trois cas se présentent :

a) 1er cas : $EQ_Y(t_1, t_2) \geq EQ_X(t_1, t_2) \geq EQ_Z(t_1, t_2)$

$[EQ_Y(t_1, t_2) \geq EQ_X(t_1, t_2) \geq EQ_Z(t_1, t_2)] \Rightarrow [EQ_{YZ}(t_1, t_2) = EQ_Z(t_1, t_2)] \wedge [EQ_{XZ}(t_1, t_2) = EQ_Z(t_1, t_2)] \Rightarrow [EQ_{YZ}(t_1, t_2) = EQ_{XZ}(t_1, t_2)] \Rightarrow [\forall\ t_1, t_2 \in \tilde{r}, [EQ_{XZ}(t_1, t_2) \geq \alpha \Rightarrow EQ_{YZ}(t_1, t_2) \geq EQ_{XZ}(t_1, t_2)]] \Rightarrow (\boldsymbol{XZ \sim>_{\alpha} YZ})$.

b) 2ème cas : $EQ_Y(t_1, t_2) \geq EQ_Z(t_1, t_2) \geq EQ_X(t_1, t_2)$

$[EQ_Y(t_1, t_2) \geq EQ_Z(t_1, t_2) \geq EQ_X(t_1, t_2)] \Rightarrow [EQ_{YZ}(t_1, t_2) = EQ_Z(t_1, t_2)] \wedge [EQ_{XZ}(t_1, t_2) = EQ_X(t_1, t_2)] \Rightarrow [EQ_{YZ}(t_1, t_2) \geq EQ_{XZ}(t_1, t_2)] \Rightarrow [\forall\ t_1, t_2 \in \tilde{r},\ [EQ_{XZ}(t_1, t_2) \geq \alpha \Rightarrow EQ_{YZ}(t_1, t_2) \geq EQ_{XZ}(t_1, t_2)]] \Rightarrow (XZ \sim>_\alpha YZ)$.

c) 3ème cas : $EQ_Z(t_1, t_2) \geq EQ_Y(t_1, t_2) \geq EQ_X(t_1, t_2)$

$[EQ_Z(t_1, t_2) \geq EQ_Y(t_1, t_2) \geq EQ_X(t_1, t_2)] \Rightarrow [EQ_{YZ}(t_1, t_2) = EQ_Y(t_1, t_2)] \wedge [EQ_{XZ}(t_1, t_2) = EQ_X(t_1, t_2)] \Rightarrow [EQ_{YZ}(t_1, t_2) \geq EQ_{XZ}(t_1, t_2)] \Rightarrow [\forall\ t_1, t_2 \in \tilde{r},\ [EQ_{XZ}(t_1, t_2) \geq \alpha \Rightarrow EQ_{YZ}(t_1, t_2) \geq EQ_{XZ}(t_1, t_2)]] \Rightarrow (\boldsymbol{XZ \sim>_\alpha YZ})$.

De **a), b)** et **c)**, on obtient : $\boldsymbol{X \sim>_\alpha Y \Rightarrow XZ \sim>_\alpha YZ}$. Par conséquent, la règle d'augmentation est valide pour une α-DFF $\boldsymbol{X \sim>_\alpha Y}$.□

FF3. Transitivité : Si $\boldsymbol{X \sim>_\alpha Y}$ et $\boldsymbol{Y \sim>_\alpha Z}$ sont valides, alors $\boldsymbol{X \sim>_\alpha Z}$ est valide.

Preuve :

Nous utilisant le lemme 1 afin de prouver la transitivité des α-DFFs.

La règle **FF3.** se réécrit en : $(X_\alpha \subseteq_f Y_\alpha) \wedge (Y_\alpha \subseteq_f Z_\alpha) \Rightarrow (X_\alpha \subseteq_f Z_\alpha)$.

- On suppose que : $(X_\alpha \subseteq_f Y_\alpha)$ et $(Y_\alpha \subseteq_f Z_\alpha)$. Cela veut dire :

  $X_\alpha \subseteq_f Y_\alpha \Leftrightarrow (\forall\ u \in X \Rightarrow u \in Y \text{ et } \mu_X(u) \leq \mu_Y(u))\ldots$ **(i)**

  $Y_\alpha \subseteq_f Z_\alpha \Leftrightarrow (\forall\ u \in Y \Rightarrow u \in Z \text{ et } \mu_Y(u) \leq \mu_Z(u))\ldots$ **(ii)**

- et on veut prouver que : $\boldsymbol{X \sim>_\alpha Z}$ $(X_\alpha \subseteq_f Z_\alpha)$

Sachant que : l'opérateur "≤" utilisé dans les expressions **(i)** et **(ii)** est l'opérateur de comparaison « inférieur ou égale », alors cet opérateur est transitif , d'où :

(i) $\wedge$ **(ii)** $\Rightarrow [(\forall\ u \in X \Rightarrow [u \in Y \Rightarrow u \in Z] \text{ et } \mu_X(u) \leq \mu_Y(u)) \leq \mu_Z(u))] \Rightarrow [\mu_X(u) \leq \mu_Z(u)]$ $\Rightarrow (X_\alpha \subseteq_f Z_\alpha) \Rightarrow (\boldsymbol{X \sim>_\alpha Z})$.

D'où : la règle de transitivité est valide pour une α-DFF $\boldsymbol{X \sim>_\alpha Y}$. □

Des propriétés additionnelles peuvent être dérivées à partir des axiomes d'Armstrong prouvés ci-dessus. Parmi ces propriétés, les trois propriétés suivantes sont valides.

FF4. Décomposition : Si $X \sim>_{\alpha} YZ$ est valide, alors $X \sim>_{\alpha} Y$ et $X \sim>_{\alpha} Z$ sont valides.

FF5. Union : Si $X \sim>_{\alpha} Y$ et $X \sim>_{\alpha} Z$ sont valide, alors $X \sim>_{\alpha} YZ$ est valide.

FF6. Pseudo-transitivité : Si $X \sim>_{\alpha} Y$ et $WY \sim>_{\alpha} Z$ sont valides, alors $WX \sim>_{\alpha} Z$ est valide.

Théorème 1 :
L'ensemble I = {FF1, FF2, FF3, FF4, FF5, FF6} forme un système complet et correct.

La preuve de ce théorème est donnée en annexe 2.

V.2.3- Processus global de découverte des α-DFFs (algorithme général α-FFD-Extract)

Avant de présenter le processus de découverte de dépendances fonctionnelles floues à partir de bases de données possibilistes, nous allons d'abord définir les différents concepts mathématiques que nous allons utiliser dans notre approche.

A)- Définitions formelles

Les définitions des différentes notions utilisées dans le processus de découverte des α-DFFs sont les suivantes.

Définition 2 : (ensemble de DFFs minimales non triviales valides à un niveau α et couverture canonique de dep($\tilde{r}$))

Un ensemble d'α-DFFs minimales non triviales, noté Φ_{α} est défini par : $\forall f \in \Phi_{\alpha} : f$ est de la forme $X \sim>_{\alpha} K$ vérifiant les conditions suivantes :

1. $X \subset \boldsymbol{R}$: X est dit partie gauche de l'α-DFF $X \sim>_{\alpha} K$. $|X| \in \{1, \ldots, n\text{-}1\}$, ($n$ étant le nombre d'attributs dans $\boldsymbol{R}$),
2. $K \in \boldsymbol{R}$, (K est dit partie droite de l'α-DFF $X \sim>_{\alpha} K$. $|K| = 1$),
3. $\forall X' \subset X : X' \sim>_{\alpha} K$ n'est pas valides dans $\tilde{\boldsymbol{r}}$,
4. $K \notin X$.

Soit dep($\tilde{r}$) l'ensemble de toutes les α-DFFs vérifiées dans la relation $\tilde{r}$. L'ensemble Φ_α constitue une couverture canonique de dep($\tilde{r}$) puisqu'à partir de cet ensemble on peut déduire toutes les α-DFFs valides dans $\tilde{r}$ en utilisant les axiomes d'Armstrong prouvées dans la section §**V.2.2**.

Définition 3 : (Ensemble des parties gauches candidates de taille *i*+1 d'un attribut *K*)
L'ensemble des parties gauches candidates, de taille (i+1), d'un attribut K, noté $L_{i+1}(\Phi_\alpha, K)$, est l'ensemble des sous-ensembles d'attributs X de taille (i+1) pouvant vérifier une α-DFFs ($X \sim>_\alpha K$). Chaque élément X de cet ensemble correspond à l'union de deux éléments X'' et X''' appartenant à $L_i(\Phi_\alpha, K)$ dont l'intersection est un sous-ensemble d'attributs X' de taille (i-1). Les sous-ensembles d'attributs X'' et X''' correspondent à des parties gauches non valides de taille i (les α-DFFs $X'' \sim>_\alpha K$ et $X''' \sim>_\alpha K$ ne sont pas valides dans $\tilde{r}$).

$$L_{i+1}(\Phi_\alpha, K) = \{X / |X| = i+1 : X = X'' \cup X''', \text{ avec } X'', X''' \in L_i(\Phi_\alpha, K), X' = X'' \cap X''' \text{ et } |X'| = i\text{-}1 \text{ et } \tilde{r} \not\models \{X'' \sim>_\alpha K,\ X''' \sim>_\alpha K\}\}.$$

Exemple : Soit $X'' = AB$ et $X''' = AC$. Alors $X = X'' \cup X''' = ABC$.

Définition 4 : (Ensemble des parties gauches de taille *i* d'un attribut *K*)
L'ensemble des parties gauches de i d'un attribut K, noté $LHS_i[K]$, est l'ensemble de sous-ensembles d'attributs X de taille i, tel que l'α-DFFs $X \sim>_\alpha K$ est valide dans $\tilde{r}$.

$$LHS_i[K] = \{X \mid |X| = i,\ \tilde{r} \models X \sim>_\alpha K\}$$

Définition 5 : (Ensemble de toutes les parties gauches valides d'un attribut *K*)
L'ensemble des parties gauches valides d'un attribut K, noté $lhs(\Phi_\alpha, K)$, correspond à l'union sur i de tous les ensembles de parties gauches de K de i. cet ensemble est défini comme suit :

$$lhs(\Phi_\alpha, K) = \{X \subset \boldsymbol{R} \mid \tilde{r} \models X \sim>_\alpha K\}$$

Définition 6 : (sous-ensemble de $\mathcal{X}_\alpha$ en accord avec $\mathcal{K}_\alpha$)
Soit $\mathcal{X}_\alpha$ la coupe de niveau α telle qu'elle est définie dans **(V.11)**, et soit $X[K]_{acc}$ un sous-ensemble de $\mathcal{X}_\alpha$ en accord $\mathcal{K}_\alpha$. $X[K]_{acc}$ est l'ensemble des couples de tuples (t_i, t_j)

appartenant à $\mathcal{X}_\alpha$, tel que (t_i, t_j) appartient à $\mathcal{K}_\alpha$ et $X\alpha_{ij} \leq K\alpha_{ij}$. En d'autres termes, $X[K]_{acc} \subseteq_f \mathcal{K}_\alpha$.

$$X[K]_{acc} = \{(t_i, t_j, X\alpha_{ij}) \in \mathcal{X}_\alpha | (t_i, t_j, K\alpha_{ij}) \in \mathcal{A}_\alpha \text{ et } X\alpha_{ij} \leq K\alpha_{ij}\}.$$

Définition 7 : (sous-ensemble de $\mathcal{X}_\alpha$ en désaccord avec $\mathcal{K}_\alpha$)

De la même manière on définit un sous-ensemble de $\mathcal{X}_\alpha$ en désaccord avec $\mathcal{K}_\alpha$, noté $X[K]_{des}$ par l'ensemble des couples de tuples (t_i, t_j) appartenant à $\mathcal{X}_\alpha$, tel que $X\alpha_{ij} > K\alpha_{ij}$.

$$X[K]_{des} = \{(t_i, t_j, X\alpha_{ij}) \in \mathcal{X}_\alpha | (t_i, t_j, K\alpha_{ij}) \in \mathcal{A}_\alpha \text{ et } X\alpha_{ij} > K\alpha_{ij}\}$$

Propriétés :

- $\mathcal{X}_\alpha = X[K]_{des} \cup X[K]_{acc}$ et $X[K]_{des} \cap X[K]_{acc} = \emptyset$,
- Si $X[K]_{acc} = \mathcal{X}_\alpha$ alors $X \sim>_\alpha K$ est valide dans $\tilde{r}$,
- $K[K]_{acc} = \mathcal{K}_\alpha$ et $K[K]_{des} = \emptyset$.

B)- Principe général du processus de découverte des α-DFFs

L'approche *stratifiée* de découverte des α-DFFs minimales non triviales valides dans une base de données possibiliste est une approche basée sur une *stratégie à partie gauche* (en anglais : Left Hand Side Strategy), ce qui signifie que nous fixons les parties droites des α-DFFs potentiellement valides et nous induisons ensuite les parties gauches correspondantes de telle sorte à avoir un ensemble d'α-DFFs valides dans la relation $\tilde{r}$. Les parties droites des α-DFFs étant réduites à un seul attribut, l'ensemble de ces parties droites est donc l'ensemble de tous les attributs de $\boldsymbol{R}$. Par contre, les parties gauches des α-DFFs peuvent être de taille entre 1 et (n-1), où n est le nombre d'attributs de $\boldsymbol{R}$. L'algorithme α-FFD-Extract est un algorithme par *niveau*, ce qui signifie que dans cet algorithme, les parties gauches $LHS_i[K]$, de taille i, d'un attribut K sont découvertes à partir des parties gauches $LHS_{i-1}[K]$ de K. Cette manière de faire permet d'élaguer l'ensemble des parties gauches candidates $L_i(\Phi_\alpha, K)$, de taille i, en éliminant lors de la construction de cet ensemble toutes les parties appartenant à l'ensemble $LHS_{i-1}[K]$.

C)- Algorithme α-FFD-Extract

Le processus complet de découverte est formalisé par l'algorithme α-FFD-Extract. Les étapes de cet algorithme peuvent être décrites de la manière suivante :

En ligne 01, l'administrateur de base de données fixe le seuil minimal de ressemblance α, entre les valeurs des attributs de la base, qu'il souhaite avoir.

En ligne 02, la procédure **ALPHA_CUT_SET** appliquée sur l'ensemble des attributs de ***R*** permet d'obtenir toutes les coupes de niveau α de ces attributs.

De la ligne 03 à la ligne 15, la boucle **For** permet de rechercher pour chaque attribut K de ***R*** ses parties gauches valides. Le procédé de recherche est le suivant : en premier lieu on initialise i à 1 (ligne 4), où i représente la taille des parties gauches recherchées, et l'ensemble $L_1(\Phi_\alpha, K)$ à l'ensemble des attributs de $\boldsymbol{R} \setminus K$ (ligne 5), ce qui empêche d'avoir des α-DFFs triviales.

Dans la boucle **While** (ligne 06 à 13) les éléments de l'ensemble $L_i(\Phi_\alpha, K)$ qui sont des parties gauches potentiellement valides d'α-DFFs seront validées par la procédure **I_LEV_LHS** (ligne 07). En ligne 8, l'ensemble $L_i(\Phi_\alpha, K)$ sera élagué en éliminant les parties gauches valides, cela nous permet d'avoir que les α-DFFs minimales. En ligne 09, l'ensemble des parties gauches candidates de taille (i+1), $L_{i+1}(\Phi_\alpha, K)$ sera construit. Cet ensemble sera élagué pour éliminer les parties gauches candidates non minimales par la procédure PRUNE (ligne 10). La variable i qui représente la taille des parties gauche est incrémentée (ligne 11), pour réitérer la boucle **While**, jusqu'à ce que l'ensemble des parties gauches candidates $L_i(\Phi_\alpha, K)$ soit vide. En ligne 13, l'ensemble de toutes les parties gauches valides des α-DFFs dans la partie droite est K est l'union sur i de toutes les parties gauches valides de taille i. Enfin, La procédure **α-FFD_OUTPUT** est une mise en forme d'α-DFFs, en faisant la correspondance entre les parties gauches et les parties droites des α-DFFs valides.

Remarque :

- Toutes les procédures évoquées précédemment (**ALPHA_CUT_SET, I_LEV_LHS**, **PRUNE, α-FFD_OUTPUT**) seront détaillées juste après l'algorithme 1 (Algorithme **α-FFD-Extract**).
- On désigne par A un attribut générique de $\boldsymbol{R}$, K un attribut partie droite, B un attribut partie gauche différente de la partie A et X un sous-ensemble d'attribut de taille supérieure à 1 représentant une partie gauche.

Algorithme 1 : Algorithme **α-FFD-Extract**

Entrée : $\tilde{r}$, relation sur R. α, seuil minimal de ressemblance des valeurs.
Sortie : Φ_α, ensemble de dépendances fonctionnelles floues minimales non triviales de niveau α valides dans $\tilde{r}$.

01: $\alpha \leftarrow v$ ($v \in]0, 1]$) ;
02: ALPHA_CUT_SET(A) ;
03: **For all** $K \in R$ **do** *//chercher les lhs des parties droites des α-DFFs*
04: $i \leftarrow 1$; *// Inialisation de la taille des parties gauche recherchées à 1*
05: $L_i(\Phi_\alpha, K) \leftarrow \{B \setminus K / B \in \boldsymbol{R}\}$; *// Inialisation de lhs candidates de K, de taille1*
06: **While** $L_i(\Phi_\alpha, K) \neq \emptyset$ **do**
07: I_LEV_LHS(K);
08: $L_i(\Phi_\alpha, K) \leftarrow L_i(\Phi_\alpha, K) \setminus LHS_i[K]$;
09: $L_{i+1}(\Phi_\alpha, K) \leftarrow \{X / |X| = i+1: X=X'' \bowtie_{X'} X''', X'', X''' \in L_i(\Phi_\alpha, A)$ et $X' = X'' \cap X'''$ et $|X'| = i-1\}$;
10: PRUNE($L_{i+1}(\Phi_\alpha, K)$) ;
11: $i \leftarrow i+1$;
12: **end while ;**
13: $lhs(\Phi_\alpha, K) \leftarrow \cup_i LHS_i[K]$;
14: **End for ;**
15: α-DFF_OUTPUT ;

D)- Détermination des coupes de niveau α

Les DFFs auxquelles nous nous intéressons sont des α-DFFs, ce qui signifie que leur validité ne s'étend pas sur l'intervalle [0, 1], mais sur un intervalle [α, 1]. Cette contrainte, malgré qu'elle restreint la plage de validité de la DFF, elle permet toute fois d'avoir une sémantique des DFFs « proche » de celle des DFs classique. Autrement dit, pour un seuil de ressemblance α proche de un "1", les DFFs sont testées sur des couples de tuples fortement ressemblants. Pour avoir les α-DFFs, nous utilisons la notion de α-coupe sur les valeurs des attributs représentées par des distributions de possibilité.

La procédure ALPHA_CUT_SET est une procédure de prétraitement[3] des données pour l'algorithme α-FFD-Extract. En ligne 01 à 03, on calcule le degré de ressemblance pour toutes les paire de tuple (t_i, t_j) de $\tilde{r}$, sur chaque attribut A de $\boldsymbol{R}$. Ce degré de ressemblance est noté $A\alpha_{ij}$, en ensuite on compare le degré de ressemblance $A\alpha_{ij}$ au seuil de ressemblance minimal fixé α. Un triplet (t_i, t_j, $A\alpha_{ij}$) appartiendra à la coupe de niveau α d'un attribut A si, et seulement si le degré de ressemblance $A\alpha_{ij}$ est supérieur ou égal à α.

L'algorithme 2 présente le pseudo-code de la procédure ALPHA_CUT_SET.

Algorithme 2: Algorithme ALPHA_CUT_SET()

Entrée : $\boldsymbol{R}$ relation
Sortie : $\mathcal{A}_\alpha$, ensemble des α-coupes

// Coupes α des attributs de la relation $\boldsymbol{R}$

```
01:   for all (t_i, t_j) ∈ r̃ × r̃  do
02:      for all  A ∈ R  do
03:         Aα_ij ← Sup_{u ∈dom} K Min (π_{ti.A}(u), π_{tj.A}(u))
04:         if   Aα_ij ≥ α   then
05:                  A_α ← A_α ∪ (t_i, t_j, Aα_ij)
06:         End if
07:      End for
08:    End for
```

E)- Validation des parties gauches de taille *i* de chaque attribut *K*

Après avoir généré des parties gauches X potentiellement valides pour une partie droite K, il est nécessaire de vérifier la validité de ces parties par rapport à K. d'après le lemme 1, une partie gauche X candidate, de taille i, appartenant à $L_i(\Phi_\alpha, K)$ est valide si, et seulement si la coupe α de X est incluse dans la coupe α de K. L'ensemble $LHS_i[K]$ des partie gauches d'un attribut K est défini donc par :

$$LHS_i[K] = \{X / \ |X| = i, \mathcal{X}_\alpha \subseteq_f \mathcal{K}_\alpha\}$$

Où : $\mathcal{X}_\alpha \subseteq_f \mathcal{K}_\alpha \Leftrightarrow \forall (t_i, t_j) \in \mathcal{X}_\alpha, (t_i, t_j) \in \mathcal{K}_\alpha$ et $X\alpha_{ij} \leq K\alpha_{ij}$

[3] Procédure de prétraitement ou étape de prétraitement dans un processus d'ECD (cf. Chapitre III).

La procédure I_LEV_LHS(K) de l'algorithme α-FFD-Extract permet la validation des parties gauche candidates en se basant sur le lemme 2 qui utilise la notion d'ensemble en désaccord avec K, notée $X[K]_{des}$ définie plus haut.

Lemme 2 :

Soit X un sous-ensemble d'attributs de $\boldsymbol{R}$, tel que $X = X' \bowtie_{X'} X''$. Vérifier si $\mathcal{X}_\alpha$ est inclus ou égal à $\mathcal{K}_\alpha$ (c'est-à-dire $X \sim\!>_\alpha K$ est valide dans $\tilde{r}$) revient à vérifier si : $X'[K]_{des} \cap_f X''[K]_{des} = \varnothing$.

Où $\cap_f$ représente l'inclusion floue de deux ensemble en utilisant la T-norme « Min ».

Formellement, on aura :

$$X \sim\!>_\alpha K \Leftrightarrow \begin{cases} X[K]_{des} = \varnothing & \textbf{si } |X| = 1 \quad \textbf{(i)} \\ X'[K]_{des} \cap_f X''[K]_{des} = \varnothing & \textbf{si } |X| > 1 \ (X = X' \bowtie_{X'} X'') \quad \textbf{(ii)} \end{cases}$$

<u>Preuve</u> :

(i) Montrer que : **(** $|X| = 1 \ \wedge X \sim\!>_\alpha K$ est valide **)** $\Leftrightarrow X[K]_{des} = \varnothing$

$X \sim\!>_\alpha K \Leftrightarrow \mathcal{X}_\alpha \subseteq_f \mathcal{K}_\alpha$

$\Leftrightarrow [X[K]_{des} \cup X[K]_{acc}] \subseteq_f [K[K]_{acc} \cup K[K]_{des}]$

$\Leftrightarrow (X[K]_{acc} \subseteq_f K[K]_{acc})$ et $(X[K]_{des} \subseteq_f K[K]_{des})$

$\Leftrightarrow X[K]_{des} \subseteq_f \varnothing$

$\Leftrightarrow X[K]_{des} = \varnothing$.

(ii) Montrer que : **(** $|X| > 1$ (i,e. $X = X' \bowtie_{X'} X'') \wedge X \sim\!>_\alpha K$ **)** $\Leftrightarrow$ **(** $X'[K]_{des} \cap_f X''[K]_{des} = \varnothing$ **)**

$X \sim\!>_\alpha K \Leftrightarrow \mathcal{X}_\alpha \subseteq_f \mathcal{K}_\alpha$

$\Leftrightarrow (\mathcal{X}''_\alpha \cap_f \mathcal{X}'''_\alpha) \subseteq_f \mathcal{K}_\alpha$

$\Leftrightarrow [X'[K]_{des} \cup X'[K]_{acc}] \cap_f [X''[K]_{des} \cup X''[K]_{acc}] \subseteq_f \mathcal{K}_\alpha$

$\Leftrightarrow \big[[X'[K]_{acc} \cap_f X''[K]_{acc}] \cup [X'[K]_{acc} \cap_f X'[K]_{des}] \cup [X'[K]_{des} \cap_f X''[K]_{acc}]$

$\cup [X'[K]_{des} \cap_f X''[K]_{des}]\big] \subseteq_f [K[K]_{acc} \cup K[K]_{des}]$

$\Leftrightarrow \quad \left[\left[[X'[K]_{acc} \cap_f X''[K]_{acc}] \cup [X'[K]_{acc} \cap_f X'[K]_{des}] \cup [X'[K]_{des} \cap_f X''[K]_{acc}]\right] \subseteq_f K[K]_{acc}\right]$ et $\left[[X'[K]_{des} \cap_f X''[K]_{des}] \subseteq_f K[K]_{des}\right]$

$\Leftrightarrow \quad [X'[K]_{des} \cap_f X''[K]_{des}] = \varnothing$. □

La procédure I_LEV_LHS fonctionne de la manière suivante : on initialise une variable x à i qui représente la taille le parties gauches candidates à vérifier (ligne 1), la variable x sert différencier la première itération des suivantes. De la ligne 02 à la ligne 18 on détermine pour chaque partie gauche candidates $X \in L_i(\Phi_\alpha, K)$ le sous-ensembles de $\mathcal{X}_\alpha$ en désaccord avec K, notés $X[K]_{des}$. Ces sous-ensembles nous serviront dans les lignes 19 à 21 à vérifier la validité des parties gauches candidates en utilisant les résultats du lemme 2.

Algorithme 3 : Algorithme I_LEV _LHS(K)

Entrée **:** $L_i(\Phi_\alpha, K)$, candidates de taille i.
Sortie **:** $LHS_i[K]$, partie gauche de taille i de l'attribut A.

```
01: x = i ;
02: if x =1
03:         then
04:             for all   X ∈ L_i(Φ_α, K)   do   // tester l'inclusion : X_α ⊆_f K_α
05:                 for all   (t_i, t_j) ∈ X_α    do
06:                     if   Xα_ij ≤ Kα_ij
07:                                 then
08:                                     X[K]_acc ← X[K]_aac ∪ (t_i, t_j, Xα_ij) ;
09:                                 else
10:                                     X[K]_des  ← X[K]_des ∪ (t_i, t_j, Xα_ij) ;
11:                     endif ;
12:                 endfor ;
13:             endfor
14:         else
15:             for all X ∈ L_{i+1}(Φ_α, K) do          //X = X'' ⋈_{X'} X''' tel que X'', X''' ∈ L_i(Φ_α,
K)
16:                             X[A]_des ← X''[A]_des ∩ X'''[A]_des ;
17:             endfor ;
18: endif ;
19: if   X_des = ∅   then
20:             LHS_i [K] ← LHS_i [K] ∪{X} ;
21: endif ;
```

F)- Elagage de l'ensemble des partie gauches candidates de taille *i*+1

Après avoir généré l'ensemble $L_{i+1}(\Phi_\alpha, K)$ des parties gauches candidates de taille (i+1) d'un attribut $K \in \boldsymbol{R}$, à partir des $L_i(\Phi_\alpha, K)$ (voir algorithme 1). Tous les éléments X de $L_{i+1}(\Phi_\alpha, K)$ représentant un sur-ensemble d'attributs d'un élément $X' \in LHS_i[K]$ (c'est-à-dire $X = X'C$, où $C \in \boldsymbol{R}$) sont éliminés (enlevés) de $L_{i+1}(\Phi_\alpha, K)$ par la procédure PRUNE. Cette procédure élague l'ensemble des parties gauches pour ne pas avoir des parties gauches non minimales. L'algorithme 4 représenté ci-après est le pseudo-code de la procédure PRUNE.

Algorithme 4 : Algorithme PRUNE($L_{i+1}(\Phi_\alpha, K)$)

Entrée: $L_{i+1}(\Phi_\alpha, K)$, ensemble de partie gauches candidates de taille i+1
Sortie: $L_{i+1}(\Phi_\alpha, K)$, ensembles de parties gauches candidates élagué

// Élaguer les parties gauches candidates non minimales

01: **for all** $X' \in LHS_i[K]$ **do**
02: **for all** $X \in L_{i+1}(\Phi_\alpha, K)$ **do**
03: **if** $X' \subset X$ **then**
04: $L_{i+1}(\Phi_\alpha, K) \leftarrow L_{i+1}(\Phi_\alpha, K) \setminus \{X\}$
05: **endif**
06: **endfor**
07: **endfor**

G)- Mise en forme des α-DFFs validées

L'obtention des α-DFFs minimales non triviale à partir des parties gauches est une simple mise en forme comme le montre l'algorithme 5 ci-après. Pour chaque attribut $K \in \boldsymbol{R}$, et pour chaque $X \in lhs(\Phi_\alpha, K)$, afficher l'$\alpha$-DFFs ($X \sim\!>_\alpha K$) valide dans la relation $\tilde{r}$ considérée.

Algorithme 5 : Algorithme α_FFD_OUTPUT

Entrée **:** $LHS(\Phi_\alpha, K)$, ensembles des parties gauches d'un attribut K.
Sortie **:** Φ_α, ensemble des α-DFFs minimales non triviales valides dans la relation $\tilde{r}$.

1: **For all** $K \in R$ **do**
2: **For all** $X \in lhs(\Phi_\alpha, K)$ **do**
3: Afficher($X \sim\!>_\alpha K$)
4: **End for**
5: $\Phi_\alpha \leftarrow \Phi_\alpha \cup (X \sim\!>_\alpha K)$
6: **End for.**

H)- Comparaison de la complexité algorithmique de α_FFD_Extract et celle de l'algorithme de WANG

Le test de la validité d'une DFF $\boldsymbol{X \sim\!> A}$ par l'algorithme de WANG prend un temps de $O[|r|^2 * (|X|+1)]$.

Dans notre cas cette complexité peut être réduite par le fait que nous effectuons une coupe de niveau α sur l'espace $|r|^2$, cela peut être expliquer comme suit :
Etant donné deux tuples quelconques $t_1, t_2 \in r$, soit P(α, X) la probabilité que $EQ_X(t_1, t_2) \geq \alpha$.

Pour un seuil de ressemblance α, la complexité de l'algorithme d'inférence des α-DFFs devient donc : $O[|r|^2 * [(|X|+1)* P(X, \alpha)]$. Tel que : $P(X, \alpha) \leq 1$.

V.3- Exemple d'application

Soient la relation *EMPLOYE*(NumEmp, NomEmp, PrenomEmp, Job, Exprerience, Salaire, NbVoit) et r une instance de la relation *EMPLOYE* à valeurs manquantes représentée comme suit :

#	NumEmp	NomEmp	PrenomEmp	Job	Exprerience	Salaire(DA)	NbVoit
1	2	Nelson	Roberto	CS	1	40 000,00	1
2	4	Young	Bruce	DR	2	60 500,00	2
3	5	Lambert	Kim	**?**	**?**	**?**	**?**
4	8	Johnson	Leslie	IN	1	25 050,00	0
5	9	Forest	Phil	IN	1	25 050,00	0
6	11	Weston	K. J.	**?**	**?**	**?**	**?**
7	12	Lee	Terri	CS	1	40 000,00	1
8	14	Young	Stewart	IN	2	28 000,00	0
9	15	Papadopoulos	Katherine	**?**	**?**	**?**	**?**
10	20	Hall	Chris	IN	1	25 050,00	0

Les abréviations CS, DR et IN représentent respectivement les valeurs du domaine Job de chef de service, directeur et ingénieur.

Par souci de concision, les attributs NumEmp, NomEmp, PrenomEmp, Job, Expérience, Salaire, NbVoit sont renommés respectivement *A, B, C, D, E, F, G.*

Le symbole **?** représente une valeur manquante (en anglais : missing value). La génération de valeurs substitutives (distributions de possibilité) pour remplacer les valeurs manquantes dans l'instance de la relation *EMPLOYE* est traitée dans ce qui suit.

V.3.1- Traitement des données manquantes (Missing Values)

Dans la section §V.1 nous avons parlé des différentes procédures de fuzzification des données imprécises qui peuvent apparaître dans une base de données floues. Les données figurant dans la table de relation précédente sont de deux types (valeurs précises et valeurs manquantes). Les transformations à effectuer sont :

- Attribution d'un degré de possibilité égal à un « 1 » pour les valeurs précises,
- Application de la transformation de Dubois et Prade afin de générer des distributions de possibilité pour remplacer les valeurs manquantes.

Application de la transformation de DUBOIS et PRADE

Pour l'attribut D :

- Les valeurs du domaine sont {CS, DR, IN}
- Le nombre de tuple à valeurs précises est : 7
- Les fréquences d'apparition des valeurs du domaine sont :

p_1= Prob(x=CS) = 2/7= 0.3 p_2 = Prob(x=DR) = 1/7= 0.15

p_3 = Prob(x=IN) = 4/7= 0.55

- La distribution de possibilité peut être générée à partir de ces probabilités comme suit :

Du moment que les probabilités sont toutes différentes, alors il y a une seule permutation possible, et qui est :

$\sigma(1) = 2, \sigma(2) = 1, \sigma(3) = 3$, à laquelle correspond $\sigma_1^{-1}(1) = 2, \sigma_1^{-1}(2) = 1, \sigma_1^{-1}(3) = 3,$

En appliquant la transformation de Dubois et Prade on obtient :

$$\pi_1 = \sum_{\{j \mid \sigma^{-1}(j) \le \sigma^{-1}(1)\}} p_j = p_1+p_2 = 0.45 ; \qquad \pi_2 = \sum_{\{j \mid \sigma^{-1}(j) \le \sigma^{-1}(2)\}} p_j = p_2 = 0.15 ;$$

$$\pi_3 = \sum_{\{j \mid \sigma^{-1}(j) \le \sigma^{-1}(3)\}} p_j = p_1+p_2, p_3 = 1.0 ;$$

La distribution de possibilité générée pour remplacer une valeur manquante pour l'attribut *D* est : « 0.45/ CS+0.15/DR+1.0/IN »

D'où : t_3.D = t_6.D = t_9.D = 0.45/ CS+0.15/DR+1.0/IN

Pour l'attribut E :

- Les valeurs du domaine sont {1, 2}
- Le nombre de tuple à valeurs précises est : 7
- Les fréquences d'apparition des valeurs du domaine, sont :

$p_1 = Prob(x = 1) = 5/7 = 0.7$ $p_2 = Prob(x = 2) = 2/7 = 0.3$

- La distribution de possibilité peut être générée à partir de ces probabilités comme suit :

Du moment que les probabilités sont différentes alors il y une seule permutation possible : $\sigma(1) = 2$, $\sigma(2) = 1$, à laquelle correspond $\sigma^{-1}(1) = 2$, $\sigma^{-1}(2) = 1$,

En appliquant la transformation de Dubois et Prade on obtient :

$\pi_1 = \sum_{\{j \mid \sigma^{-1}(j) \leq \sigma^{-1}(1)\}} p_j = p_1+p_2= 1$; $\pi_2 = \sum_{\{j \mid \sigma^{-1}(j) \leq \sigma^{-1}(2)\}} p_j = p_2 = 0.3$;

La distribution de possibilité générée pour remplacer une valeur manquante pour l'attribut E est : « 1.0/1+0.3/2 »

D'où : $t_3.E = t_6.E = t_9.E = 1.0/1+0.3/2$

Pour l'attribut F :

- Les valeurs du domaine sont {25050,00 ; 28 000,00 ; 40000,00 ; 60 500,00}
- Le nombre de tuple à valeurs précises est : 7
- La fréquence d'apparition des valeurs du domaine sont :

$p_1 = Prob(x = 25050,00) = 3/7= 0.4$ $p_2 = Prob(x = 28000,00) = 1/7= 0.15$

$p_3 = Prob(x = 400500,00) = 2/7= 0.3$ $p_4 = Prob(x = 60500,00) = 1/7= 0.15.$

- La distribution de possibilité peut être générée à partir de ces probabilités comme suit :

Du moment que $p_2 = p_4$ il y a donc deux permutations possibles :

- p_2 est le plus petit : $\sigma_1(1) = 2$, $\sigma_1(2) = 4$, $\sigma_1(3) = 3$, $\sigma_1(4) = 1$, à laquelle correspond $\sigma_1^{-1}(1) = 4$, $\sigma_1^{-1}(2) = 1$, $\sigma_1^{-1}(3) = 3$, $\sigma_1^{-1}(4) = 2$;
- p_4 est le plus petit : $\sigma_2(1) = 4$, $\sigma_2(2) = 2$, $\sigma_2(3) = 3$, $\sigma_2(4) = 1$ à laquelle correspond $\sigma_2^{-1}(1) = 4$, $\sigma_2^{-1}(2) = 2$, $\sigma_2^{-1}(3) = 3$, $\sigma_2^{-1}(4) = 1$.

En appliquant la transformation de Dubois et Prade on obtient :

$\pi_1 = \text{Max}_{l=1,L} \sum_{\{j \mid \sigma l^{-1}(j) \le \sigma l^{-1}(1)\}} p_j = \text{Max}(\sum_{\{j \mid \sigma 1^{-1}(j) \le \sigma 1^{-1}(1)\}} p_j, \sum_{\{j \mid \sigma 2^{-1}(j) \le \sigma 2^{-1}(1)\}} p_j) =$

$\text{Max}(p_1+p_2+p_3+p_4, p_1+p_2+p_3+p_4) = p_1+p_2+p_3+p_4 = 1.0$;

$\pi_2 = \text{Max}_{l=1,L} \sum_{\{j \mid \sigma l^{-1}(j) \le \sigma l^{-1}(2)\}} p_j = \text{Max}(\sum_{\{j \mid \sigma 1^{-1}(j) \le \sigma 1^{-1}(2)\}} p_j, \sum_{\{j \mid \sigma 2^{-1}(j) \le \sigma 2^{-1}(2)\}} p_j) =$

$\text{Max}(p_2, p_2+ p_4) = p_2+ p_4 = 0.30$;

$\pi_3 = \text{Max}_{l=1,L} \sum_{\{j \mid \sigma l^{-1}(j) \le \sigma l^{-1}(3)\}} p_j = \text{Max}(\sum_{\{j \mid \sigma 1^{-1}(j) \le \sigma 1^{-1}(3)\}} p_j, \sum_{\{j \mid \sigma 2^{-1}(j) \le \sigma 2^{-1}(3)\}} p_j) =$

$\text{Max}(p_2+p_3+p_4, p_2+p_3+p_4) = p_2+p_3+p_4 = 0.60$;

$\pi_4 = \text{Max}_{l=1,L} \sum_{\{j \mid \sigma l^{-1}(j) \le \sigma l^{-1}(4)\}} p_j = \text{Max}(\sum_{\{j \mid \sigma 1^{-1}(j) \le \sigma 1^{-1}(4)\}} p_j, \sum_{\{j \mid \sigma 2^{-1}(j) \le \sigma 2^{-1}(4)\}} p_j) =$

$\text{Max}(p_2+p_4, p_4) = \text{Max}(0.30, 0.15) = 0.30$;

La distribution de possibilité générée pour remplacer une valeur manquante pour l'attribut F est : « 1.0/25050+0.3/28 000 +0.60/40000+0.30/60500 »

D'où : t_3.E = t_6.E = t_9.E = 1.0/25050+0.3/28 000 +0.60/40000+0.30/60500

Pour l'attribut G :

- Les valeurs du domaine sont {0 ; 1 ; 2}
- Le nombre de tuple à valeurs précises est : 7
- Les fréquences d'apparition des valeurs du domaine sont :

$p_1 = \text{Prob}(x = 0) = 4/7 = 0.55$ $p_2 = \text{Prob}(x = 1) = 2/7 = 0.3$

$p_3 = \text{Prob}(x = 2) = 1/7 = 0.15$

- La distribution de possibilité peut être générée à partir de ces probabilités comme suit :

Du moment que les probabilités sont différentes alors il y aune seule permutation possible : $\sigma(1) = 3$, $\sigma(2) = 2$, $\sigma_1(3) = 1$, à laquelle correspond $\sigma_1^{-1}(1) = 3$, $\sigma_1^{-1}(2) = 2$, $\sigma_1^{-1}(3) = 1$;

En appliquant la transformation de Dubois et Prade on obtient :

$\pi_1 = \sum_{\{j \mid \sigma^{-1}(j) \le \sigma^{-1}(1)\}} p_j = p_1+p_2+p_3 = 1.0$; $\pi_2 = \sum_{\{j \mid \sigma^{-1}(j) \le \sigma^{-1}(2)\}} p_j = p_2+ p_3 = 0.45$;

$\pi_3 = \sum_{\{j \mid \sigma^{-1}(j) \le \sigma^{-1}(3)\}} p_j = p_3 = 0.15$;

La distribution de possibilité générée pour remplacer une valeur manquante pour l'attribut F est : « 1.0/0+0.45/1+0.15/2 »

D'où : t_3.E = t_6.E = t_9.E = 1.0/0+0.45/1+0.15/2

Base de données après fuzzification :

A l'issue de l'étape de consolidation dans un processus de KDD, l'instance de la relation *EMPLOYE* que nous obtenant après fuzzification et mise en forme est la suivante :

#	**Numemp**	**Nomemp**	**Prenomemp**	**Job**	**Expérience**	**Salaire** (DA)	**Nbvoit**
1	1.0/2	1.0/Nelson	1.0/Roberto	1.0/CS	1.0/1	1/40 000	1.0/1
2	1.0/4	1.0/Young	1.0/Bruce	1.0/DR	1.0/2	1/55 500	1.0/2
3	1.0/5	1.0/Lambert	1.0/Kim	0.45/CS+0.15/DR+1.0/IN	1.0/1+0.3/2	1.0/25050+0.3/28000+ 0.60/40000+0.30/55500	1.0/0+0.45/1+0.15/2
4	1.0/8	1.0/Johnson	1.0/Leslie	1.0/IN	1.0/1	1/25 050	1.0/0
5	1.0/9	1.0/Forest	1.0/Phil	1.0/IN	1.0/1	1/25 050	1.0/0
6	1.0/11	1.0/Weston	1.0/K. J.	0.45/CS+0.15/DR+1.0/IN	1.0/1+0.3/2	1.0/25050+0.3/28000+ 0.60/40000+0.30/55500	1.0/0+0.45/1+0.15/2
7	1.0/12	1.0/Lee	1.0/Terri	1.0/CS	1.0/1	1/45 332	1.0/1
8	1.0/14	1.0/Young	1.0/Stewart	1.0/IN	1.0/2	1/30 482	1.0/0
9	1.0/15	1.0/Papadopoulos	1.0/Katherine	0.45/ CS+0.15/DR+1.0/IN	1.0/1+0.3/2	1.0/25050+0.3/28000+ 0.60/40000+0.30/55500	1.0/0+0.45/1+0.15/2
10	1.0/20	1.0/Hall	1.0/Chris	1.0/IN	1.0/1	1/25 050	1.0/0

Cette base de données peut être soumise à présent à notre algorithme de Data Mining α-FFD-Extract afin d'inférer les α-DFFs minimales non triviales valides dans cette instance de la base.

Nous avons choisi pour exemple d'inférer des α-DFFs, par l'algorithme α-FFD-Extract, de deux niveaux α et qui sont : $\alpha = 0.3$ et $\alpha = 1.0$
Les résultats obtenus sont donnés comme suit :

a) Découverte des α-DFFs minimales non triviales à un niveau α = 0.3 :

I)- Les coupes de niveau α (Procédure ALPHA_CUT_SET)

$A_\alpha = \varnothing$;

B_α = {[(2, 8); 1]};

$C_\alpha = \varnothing$;

D_α = {[(1, 7) ; 1] ; [(3, 4) ;1] ; [(3, 5) ;1] ; [(3, 6) ;1]; [(3, 8) ;1]; [(3, 9) ;1]; [(3, 10) ;1]; [(4, 5) ;1]; [(4, 6) ;1]; [(4, 8) ;1] ; [(4, 9) ;1] ; [(4, 10) ;1] ; [(5, 6) ;1] ; [(5, 8) ;1] ; [(5, 9) ;1] ; [(5, 10) ;1] ; [(6, 8) ;1] ; [(6, 9) ;1] ; [(6, 10) ;1] ; [(8, 9) ;1] ; [(8, 10) ;1] ; [(9, 10) ;1] ; [(1, 3) ;0.45] ; [(1, 6) ; 0.45] ; [(1, 9) ; 0.45] ; [(3, 7) ; 0.45] ; [(6, 7) ; 0.45] ; [(7, 9) ; 0.45]}

E_α = {[(2, 8) ; 1] ; [(1, 3) ; 1] ; [(1, 4) ; 1] ; [(1, 5) ; 1] ; [(1, 6) ;1] ; [(1, 7) ;1] ; [(1, 9) ;1]; [(1, 10) ;1]; [(3, 4) ;1]; [(3, 5) ;1]; [(3, 6) ;1]; [(3, 7) ;1]; [(3, 9) ;1]; [(3, 10) ;1]; [(4, 5) ;1]; [(4, 6) ;1]; [(4, 7) ;1]; [(4, 9) ;1] ; [(4, 10) ;1] ; [(5, 6) ;1] ; [(5, 7) ;1]; [(5, 9) ;1] ; [(5, 10) ;1] ; [(6, 7) ;1]; [(6, 9) ;1] ; [(6, 10) ;1] ; [(9, 10) ;1] ; [(7, 9) ;1]; [(7, 10) ;1]; [(9, 10) ;1] ; [(2, 3) ;0.3] ; [(2, 6) ;0.3] ; [(2, 9) ; 0.3] ; [(3, 8) ; 0.3] ; [(6, 8) ;0.3] ; [(8, 9) ;0.3]}

F_α = {[(1, 7) ; 1] ; [(3, 4) ;1] ; [(3, 5) ;1] ; [(3, 6) ;1]; [(3, 9) ;1]; [(3, 10) ;1]; [(4, 5) ;1] ; [(4, 6) ;1]; [(4, 9) ;1] ; [(4, 10) ;1] ; [(5, 6) ;1] ; [(5, 9) ;1] ; [(5, 10) ;1] ; [(6, 9) ;1] ; [(6, 10) ;1] ; [(9, 10) ;1] ; [(1, 3) ;0.6] ; [(1, 6) ;0.6] ; [(1, 9) ;0.6] ; [(2, 3) ;0.3] ; [(2, 6) ;0.3] ; [(2, 9) ;0.3] ; [(3, 8) ;0.3] ; [(3, 7) ;0.6] ; [(6, 7) ;0.6] ; [(6, 8) ;0.3] ; [(7, 9) ;0.6] ; [(8, 9) ;0.3]}

G_α = {[(1, 7) ; 1] ; [(3, 4) ;1] ; [(3, 5) ;1] ; [(3, 6) ;1]; [(3, 8) ;1]; [(3, 9) ;1]; [(3, 10) ;1]; [(4, 5) ;1]; [(4, 6) ;1]; [(4, 8) ;1] ; [(4, 9) ;1] ; [(4, 10) ;1] ; [(5, 6) ;1] ; [(5, 8) ;1] ; [(5, 9) ;1] ; [(5, 10) ;1] ; [(6, 8) ;1] ; [(6, 9) ;1] ; [(6, 10) ;1] ; [(8, 9) ;1] ; [(8, 10) ;1] ; [(9, 10) ;1] ; [(1, 3) ;0.45] ; [(1, 6) ; 0.45] ; [(1, 9) ; 0.45] ; [(3, 7) ; 0.45] ; [(6, 7) ; 0.45] ; [(7, 9) ; 0.45]}

II)- Parties gauches des attributs (Procédure I_LEV_LHS)

Pour l'attribut *A* :

$L_1(\Phi_\alpha, A) = \{ B, C, D, E, F, G\}$

	$X[A]_{acc}$	$X[A]_{des}$
B	$\varnothing$	$\mathcal{B}_\alpha$
C	C_α	$\varnothing$
D	$\varnothing$	$\mathcal{D}_\alpha$
E	$\varnothing$	$\mathcal{E}_\alpha$
F	$\varnothing$	$\mathcal{F}_\alpha$
G	$\varnothing$	$\mathcal{G}_\alpha$

$LHS_1[A] = \{C\}$
$L_1(\Phi_\alpha, A) = \{ B, D, E, F, G\}$
$L_2(\Phi_\alpha, A) = \{BD, BE, BF, BG, DE, DF, DG, EF, EG, FG\}$

	$\mathcal{X}_\alpha$	$X[A]_{acc}$	$X[A]_{des}$
BD	$\varnothing$	$\mathcal{BD}_\alpha$	$\varnothing$
BE	$\mathcal{B}_\alpha$	$\varnothing$	$\mathcal{B}_\alpha$
BF	$\varnothing$	$\mathcal{BF}_\alpha$	$\varnothing$
BG	$\varnothing$	$\mathcal{BG}_\alpha = \mathcal{BD}_\alpha$	$\varnothing$
DE	$\mathcal{D}_\alpha \cup$ {[(3, 8) ;0.3] ;[(8, 9) ;0.3]} - {[(3, 8) ;1]; [(4, 8) ;1] ; [(5, 8) ;1] ; [(6, 8) ;1] ; [(8, 9) ;1] ; [(8, 10) ;1]}	$\varnothing$	$\mathcal{DE}_\alpha$
DF	$\mathcal{F}_\alpha \cup$ {[(1, 3) ;0.45] ; [(1, 6) ; 0.45] ; [(1, 9) ; 0.45] ; [(3, 7) ; 0.45] ; [(6, 7) ; 0.45] ; [(7, 9) ; 0.45]} - {[(1, 3) ;0.6] ; [(1, 6) ;0.6] ; [(1, 9) ;0.6] ; [(2, 3) ;0.3] ; [(2, 6) ;0.3] ; [(2, 9) ;0.3] ; [(3, 7) ;0.6] ; [(6, 7) ;0.6] ; [(7, 9) ;0.6]}	$\varnothing$	$\mathcal{DF}_\alpha$
DG	$\mathcal{D}_\alpha$	$\varnothing$	$\mathcal{D}_\alpha$
EF	$\mathcal{F}_\alpha$	$\varnothing$	$\mathcal{EF}_\alpha$
EG	$\mathcal{G}_\alpha \cup$ {[(3, 8) ;0.3] ;[(8, 9) ;0.3]} - {[(3, 8) ;1]; [(4, 8) ;1] ; [(5, 8) ;1] ; [(6, 8) ;1] ; [(8, 9) ;1] ; [(8, 10) ;1]}	$\varnothing$	$\mathcal{DG}_\alpha = \mathcal{EG}_\alpha$
FG	$\mathcal{F}_\alpha \cup$ {[(1, 3) ;0.45] ; [(1, 6) ; 0.45] ; [(1, 9) ; 0.45] ; [(3, 7) ; 0.45] ; [(6, 7) ; 0.45] ; [(7, 9) ; 0.45]} - {[(1, 3) ;0.6] ; [(1, 6) ;0.6] ; [(1, 9) ;0.6] ; [(2, 3) ;0.3] ; [(2, 6) ;0.3] ; [(2, 9) ;0.3] ; [(3, 7) ;0.6] ; [(6, 7) ;0.6] ; [(7, 9) ;0.6]}	$\varnothing$	$\mathcal{FG}_\alpha = \mathcal{DF}_\alpha$

$LHS_2[A] = \{BD, BF, BG\}$
$L_2(\Phi_\alpha, A) = \{DE, BE, DF, DG, EF, EG, FG \}$
$L_3(\Phi_\alpha, A) = \{ DEF, DEG, DFG, EFG, BDE, BEF, BEG\}$
(Procédure PRUNE**).** Après élagage : $L_3(\Phi_\alpha, A) = \{ DEF, DEG, DFG, EFG\}$

	$\mathcal{X}_\alpha$	$X[A]_{acc}$	$X[A]_{des}$
DEF	$\mathcal{DE}_\alpha$	$\varnothing$	$\mathcal{DE}_\alpha$
DEG	$\mathcal{DE}_\alpha$	$\varnothing$	$\mathcal{DE}_\alpha$
DFG	$\mathcal{DF}_\alpha$	$\varnothing$	$\mathcal{DF}_\alpha$
EFG	$\mathcal{DE}_\alpha$	$\varnothing$	$\mathcal{EFG}_\alpha$

$LHS_3[A] = \varnothing$
$L_3(\Phi_\alpha, A) = \{ DEF, DEG, DFG, EFG\}$
$L_4(\Phi_\alpha, A) = \{ DEFG\}$

	$\mathcal{X}_\alpha$	$X[A]_{acc}$	$X[A]_{des}$
DEFG	$\mathcal{DE}_\alpha$	$\varnothing$	$\mathcal{DE}_\alpha$

$LHS_4[A] = \varnothing$
$L_4(\Phi_\alpha, A) = \{ DEFG\}$
$L_5(\Phi_\alpha, A) = \varnothing$

$lhs((\Phi_\alpha, A) = \{C, BD, BE, BF, BG\}$

Pour l'attribut *B* :

$L_1(\Phi_\alpha, B) = \{A, C, D, E, F, G\}$

$lhs((\Phi_\alpha, B) = \{C, A\}$

Pour l'attribut *C* :

$L_1(\Phi_\alpha, C) = \{A, B, D, E, F, G\}$

$lhs((\Phi_\alpha, C) = \{A, BD, BE, BF, BG\}$

Pour l'attribut *D* :

$L_1(\Phi_\alpha, D) = \{A, B, C, E, F, G\}$

$lhs((\Phi_\alpha, D) = \{A, C, BF, G\}$

Pour l'attribut *E* :

$L_1(\Phi_\alpha, E) = \{A, B, C, D, F, G\}$

$lhs((\Phi_\alpha, E) = \{A, C, B, F\}$

Pour l'attribut *F* :

$L_1(\Phi_\alpha, F) = \{A, B, C, D, E, G\}$

$lhs((\Phi_\alpha, F) = \{A, C, BD, BG, DE, EG\}$

Pour l'attribut *G* :

$L_1(\Phi_\alpha, G) = \{A, B, C, D, E, F\}$

$lhs((\Phi_\alpha, D) = \{A, C, BF, D\}$

III)- Les α_DFFs minimales non triviales valides dans *r*

(Procédure α_FFD_OUTPUT)

$\Phi_\alpha = \{$ $C \sim>_{0.3} A$, $BG \sim>_{0.3} A$, $BF \sim>_{0.3} A$, $BE \sim>_{0.3} A$,
$BD \sim>_{0.3} A$, $C \sim>_{0.3} B$, $A \sim>_{0.3} B$, $BG \sim>_{0.3} C$,
$BF \sim>_{0.3} C$, $BE \sim>_{0.3} C$, $BD \sim>_{0.3} C$, $A \sim>_{0.3} C$,
$G \sim>_{0.3} D$, $BF \sim>_{0.3} D$, $C \sim>_{0.3} D$, $A \sim>_{0.3} D$,
$F \sim>_{0.3} E$, $B \sim>_{0.3} E$, $C \sim>_{0.3} E$, $A \sim>_{0.3} E$,
$EG \sim>_{0.3} F$, $DE \sim>_{0.3} F$, $BG \sim>_{0.3} F$, $BD \sim>_{0.3} F$,
$C \sim>_{0.3} F$, $A \sim>_{0.3} F$, $A \sim>_{0.3} G$, $C \sim>_{0.3} G$,
$D \sim>_{0.3} G$, $BF \sim>_{0.3} G$ $\}$.

b) Découverte des α-DFFs minimales non triviales à un niveau α = 1.0 :

I)- Les coupes de niveau α (Procédure ALPHA_CUT_SET)

$A_\alpha = \varnothing$;

B_α = {[(2, 8); 1]};

$C_\alpha = \varnothing$;

D_α = {[(1, 7) ; 1] ; [(3, 4) ;1] ; [(3, 5) ;1] ; [(3, 6) ;1]; [(3, 8) ;1]; [(3, 9) ;1]; [(3, 10) ;1]; [(4, 5) ;1]; [(4, 6) ;1]; [(4, 8) ;1] ; [(4, 9) ;1] ; [(4, 10) ;1] ; [(5, 6) ;1] ; [(5, 8) ;1] ; [(5, 9) ;1] ; [(5, 10) ;1] ; [(6, 8) ;1] ; [(6, 9) ;1] ; [(6, 10) ;1] ; [(8, 9) ;1] ; [(8, 10) ;1] ; [(9, 10) ;1]};

E_α = {[(2, 8) ; 1] ; [(1, 3) ; 1] [(1, 4) ; 1] ; [(1, 5) ; 1] ; [(1, 6) ;1] ; [(1, 7) ;1] ; [(1, 9) ;1]; [(1, 10) ;1]; [(3, 4) ;1]; [(3, 5) ;1]; [(3, 6) ;1]; [(3, 7) ;1]; [(3, 9) ;1]; [(3, 10) ;1]; [(4, 5) ;1]; [(4, 6) ;1]; [(4, 7) ;1]; [(4, 9) ;1] ; [(4, 10) ;1] ; [(5, 6) ;1] ; [(5, 7) ;1]; [(5, 9) ;1] ; [(5, 10) ;1] ; [(6, 7) ;1]; [(6, 9) ;1] ; [(6, 10) ;1] ; [(9, 10) ;1] ; [(7, 9) ;1]; [(7, 10) ;1]; [(9, 10) ;1];}

F_α = {[(1, 7) ; 1] ; [(3, 4) ;1] ; [(3, 5) ;1] ; [(3, 6) ;1]; [(3, 9) ;1]; [(3, 10) ;1]; [(4, 5) ;1]; [(4, 6) ;1]; [(4, 9) ;1] ; [(4, 10) ;1] ; [(5, 6) ;1] ; [(5, 9) ;1] ; [(5, 10) ;1] ; [(6, 9) ;1] ; [(6, 10) ;1] ; [(9, 10) ;1]}

G_α = {[(1, 7) ; 1] ; [(3, 4) ;1] ; [(3, 5) ;1] ; [(3, 6) ;1]; [(3, 8) ;1]; [(3, 9) ;1]; [(3, 10) ;1]; [(4, 5) ;1]; [(4, 6) ;1]; [(4, 8) ;1] ; [(4, 9) ;1] ; [(4, 10) ;1] ; [(5, 6) ;1] ; [(5, 8) ;1] ; [(5, 9) ;1] ; [(5, 10) ;1] ; [(6, 8) ;1] ; [(6, 9) ;1] ; [(6, 10) ;1] ; [(8, 9) ;1] ; [(8, 10) ;1] ; [(9, 10) ;1]}

II)- Parties gauches des attributs (Procédure I_LEV_LHS)

Pour l'attribut *A* :

$L_1(\Phi_\alpha, A) = \{ B, C, D, E, F, G\}$

	$X[A]_{acc}$	$X[A]_{des}$
B	$\varnothing$	B_α
C	C_α	$\varnothing$
D	$\varnothing$	D_α
E	$\varnothing$	E_α
F	$\varnothing$	F_α
G	$\varnothing$	G_α

$LHS_1[A] = \{C\}$
$L_1(\Phi_\alpha, A) = \{ B, D, E, F, G\}$
$L_2(\Phi_\alpha, A) = \{BD, BE, BF, BG, DE, DF, DG, EF, EG, FG\}$

	$\mathcal{X}_\alpha$	$X[A]_{acc}$	$X[A]_{des}$
BD	$\varnothing$	BD_α	$\varnothing$
BE	$\mathcal{B}_\alpha$	$\varnothing$	$\mathcal{B}_\alpha$
BF	$\varnothing$	$\mathcal{BF}_\alpha$	$\varnothing$
BG	$\varnothing$	$\mathcal{BG}_\alpha$	$\varnothing$
DE	$\mathcal{E}_\alpha$	$\varnothing$	E_α
DF	$\mathcal{F}_\alpha$	$\varnothing$	F_α
DG	$\mathcal{D}_\alpha$	$\varnothing$	D_α
EF	$\mathcal{E}_\alpha$	$\varnothing$	E_α
EG	$\mathcal{E}_\alpha$	$\varnothing$	E_α
FG	$\mathcal{F}_\alpha$	$\varnothing$	F_α

$LHS_2[A] = \{BD, BF, BG\}$
$L_2(\Phi_\alpha, A) = \{DE, BE, DF, DG, EF, EG, FG\}$
$L_3(\Phi_\alpha, A) = \{DEF, DEG, DFG, EFG, BDE, BEF, BEG\}$
(Procédure PRUNE**).** Après élagage on aura : $L_3(\Phi_\alpha, A) = \{DEF, DEG, DFG, EFG\}$

	$\mathcal{X}_\alpha$	$X[A]_{acc}$	$X[A]_{des}$
DEF	$\mathcal{E}_\alpha$	$\varnothing$	$\mathcal{E}_\alpha$
DEG	$\mathcal{E}_\alpha$	$\varnothing$	$\mathcal{E}_\alpha$
DFG	$\mathcal{F}_\alpha$	$\varnothing$	$\mathcal{F}_\alpha$
EFG	$\mathcal{F}_\alpha$	$\varnothing$	$\mathcal{F}_\alpha$

$LHS_3[A] = \varnothing$
$L_3(\Phi_\alpha, A) = \{DEF, DEG, DFG, EFG\}$
$L_4(\Phi_\alpha, A) = \{DEFG\}$

	$\mathcal{X}_\alpha$	$X[A]_{acc}$	$X[A]_{des}$
$\mathcal{DEFG}$	$\mathcal{E}_\alpha$	$\varnothing$	$\mathcal{E}_\alpha$

$LHS_4[A] = \varnothing$
$L_4(\Phi_\alpha, A) = \{DEFG\}$
$L_5(\Phi_\alpha, A) = \varnothing$

$$lhs((\Phi_\alpha, A) = \{C, BD, BE, BF, BG\}$$

Pour l'attribut *B* :

$$L_1(\Phi_\alpha, B) = \{A, C, D, E, F, G\}$$

$$lhs((\Phi_\alpha, B) = \{C, A\}$$

Pour l'attribut *C* :

$$L_1(\Phi_\alpha, C) = \{A, B, D, E, F, G\}$$

$$lhs((\Phi_\alpha, C) = \{A, BD, BE, BF, BG\}$$

Pour l'attribut *D* :

$L_1(\Phi_\alpha, D) = \{A, B, C, E, F, G\}$

$lhs((\Phi_\alpha, D) = \{A, C, F, G\}$

Pour l'attribut *E* :

$L_1(\Phi_\alpha, E) = \{A, B, C, D, F, G\}$

$lhs((\Phi_\alpha, E) = \{A, C, B, F\}$

Pour l'attribut *F* :

$L_1(\Phi_\alpha, F) = \{A, B, C, D, E, G\}$

$lhs((\Phi_\alpha, F) = \{A, C, BD, BG, DE, EG\}$

Pour l'attribut *G* :

$L_1(\Phi_\alpha, G) = \{A, B, C, D, E, F\}$

$lhs((\Phi_\alpha, G) = \{A, C, F, D\}$

III)- Les α_DFFs minimales non triviales valides dans *r*

(Procédure α_FFD_OUTPUT)

$\Phi_\alpha = \{$ $C \sim>_{1.0} A$, $BG \sim>_{1.0} A$, $BF \sim>_{1.0} A$, $BE \sim>_{1.0} A$,
$BD \sim>_{1.0} A$, $C \sim>_{1.0} B$, $A \sim>_{1.0} B$, $BG \sim>_{1.0} C$,
$BF \sim>_{1.0} C$, $BE \sim>_{1.0} C$, $BD \sim>_{1.0} C$, $A \sim>_{1.0} C$,
$G \sim>_{1.0} D$, $F \sim>_{1.0} D$, $C \sim>_{1.0} D$, $A \sim>_{1.0} D$,
$F \sim>_{1.0} E$, $B \sim>_{1.0} E$, $C \sim>_{1.0} E$, $A \sim>_{1.0} E$,
$EG \sim>_{1.0} F$, $DE \sim>_{1.0} F$, $BG \sim>_{1.0} F$, $BD \sim>_{1.0} F$,
$C \sim>_{1.0} F$, $A \sim>_{1.0} F$, $A \sim>_{1.0} G$, $C \sim>_{1.0} G$,
$D \sim>_{1.0} G$, $F \sim>_{1.0} G$ $\}$

c) Analyse des résultats obtenus dans les deux cas :

- La première constatation est le fait que les α-DFFs valides au niveau $\alpha = 0.3$ restent valide au niveau $\alpha = 1.0$,

- Les α-DFFs générées peuvent être validées par le DBA en les comparant aux DFs fixées lors de la conception de la base de données.

- L'algorithme α-FFD-Extract doit être exécuté à chaque fois que la taille de la base de données aurait augmentée de manière significative. Par exemple, après augmentation de 15%, les distributions de possibilité doivent être reconsidérées en appliquant la

transformation de Dubois et Prade ou une autre méthode de génération de distributions de possibilité. En suite, l'algorithme α-FFD-Extract sera exécuté à nouveau pour vérifier si les α-DFFs validées auparavant restent toujours vérifiées dans la base de données.

V.4- Conclusion

Dans ce chapitre, nous avons présenté un processus complet d'Extraction de Connaissances à partir des Données relatif à l'inférence des dépendances fonctionnelles floues dans des bases de données possibilistes. Nous avons expliqué en premier lieu l'étape de préparation des données de la base. Ensuite, nous avons présenté une approche dite *stratifiée* des DFFs qui consiste en une nouvelle définition de la notion de DFF, appelée α-DFF, et la preuve d'un ensemble de règles d'inférence des α-DFFs valides dans une base de données possibiliste. Le procédé Data Mining pour la découverte des α-DFFs est formalisé par l'algorithme α-FFD-Extract. Enfin, nous avons présenté dans la section **§V.4** de ce chapitre les résultats d'application du processus de découverte des α-DFFs sur un exemple de relation représentée par une instance de la relation *EMPLOYE*.

CONCLUSION ET PERSPECTIVES

Bilan et Contributions

Notre travail a porté sur le problème de découverte des dépendances fonctionnelles floues dans des bases de données imprécises.

Le premier problème auquel nous nous sommes confrontés est la variété de définitions de la notion de dépendance fonctionnelle floue. Un choix s'imposait alors dans ce cas. Après étude des différentes DFFs existantes dans la littérature, nous avons constaté que certaines définitions sont saines (respect de la règle implicative) tandis que d'autres respectent la sémantique d'une DFs classique (i,e. si la ressemblance dans la partie gauche est proche de 1, la ressemblance dans la partie gauche est assez conséquente). Ces deux caractéristiques, à notre avis, sont aussi importantes l'une que l'autre pour une DFF. Par conséquent, nous avons proposé une nouvelle définition de DFF, regroupant les deux caractéristiques, appelée α-DFF, et nous avons prouvé les axiomes d'Armstrong correspondants. Ce qui nous permet d'inférer les α-DFFs minimales non triviales seulement, puisque les autres pourront être déduites par ces axiomes.

Après avoir proposé une définition de DFF en l'occurrence l'α-DFF, nous avons proposé une approche de découverte de ces α-DFFs à partir de bases de données possibilistes. Notre première idée a été d'adapter l'algorithme Dep-miner à l'inférence des DFFs. Le problème qu'on a rencontré est la *non transitivité* de la relation de ressemblance $EQ_X(t_i, t_j)$ [cf. formule **(IV.41)**], ce qui empêche d'avoir les classes d'équivalences des tuples, et par conséquent des partitions sur $\tilde{r}$. La généralisation de l'algorithme Dep-miner déjà existant devenait impossible. Nous avons alors proposé un nouvel algorithme nommé α-FFD-Extract. Les apports de cet algorithme peuvent être résumés ainsi :

- **Test de la validité des α-DFFs :**

Pour tester la validité des α-DFFs, notées $X \sim>_{\alpha} Y$, l'algorithme utilise les coupes de niveau α des sous-ensembles d'attributs X et Y. Le lemme 1 dans le chapitre **V** indique la condition de validité d'une α-DFF. Cette validité est exprimée par l'inclusion floue de la coupe de niveau α de la partie gauche, notée X_{α}, dans la coupe de niveau α de la partie droite, notée de Y_{α}. La procédure **I_LEV_LHS** permet de valider les parties gauches par rapport aux parties droites des α-DFFs.

- **Elagage :**

L'élagage se situe à deux niveaux dans notre algorithme. Premièrement**,** on élague les tuples en utilisant la notion de coupe de niveau α sur l'ensemble des couples de tuples de la base de données pour ne sélectionner que les couple tuples fortement ressemblants (leur degré de ressemblance est proche de 1) pour tout attribut de la relation. Ensuite, on élague les attributs, en supprimant d'un coup un ensemble d'attributs, et tous ses sur-ensembles. Tel que, tout les sur-ensemble Y d'un ensemble d'attributs X (X appartient à l'ensemble des parties gauches, de taille i, valides pour un attribut K) seront élagués du fait que les DFFs recherchées sont des α-DFFs minimales non triviales.

Par la suite, nous avons illustré à travers un exemple détaillé toutes les étapes de notre approche (préparation des données, fuzzification, inférence des α-DFFs.

Perspectives

Les travaux réalisés dans ce mémoire ouvrent diverses perspectives. Nous en présentons ci-dessous quelques unes.

Du point de vue pratique, il est intéressant d'avoir une base de données possibiliste réelle sur la quelle effectuer les tests de notre approche. Pour le moment nous n'avons pas pu trouver une base de données possibiliste pour nos tests. Ce que nous projetons de faire et de continuer à travailler sur l'inférence des distributions de possibilités pour des valeurs manquantes en adoptant d'autres méthodes peut être plus performantes que la transformation de Dubois et Prade qui reste tout de même limitée au cas discret et à des valeurs empiriques.

En ce qui concerne notre algorithme α-FFD-Extract, son implémentation est en cours de réalisation.

Ce qui peut être aussi intéressant à faire est de garder trace de l'exécution de l'outil d'inférence et d'ajouter d'autres procédures permettant de ne pas ré-exécuter la totalité des procédures à chaque fois qu'une nouvelle insertion dans une relation aura lieu, ce qui fait que le coût de calcul du nouvel ensemble de DFFs valides sera réduit.

Mais de manière plus tranchée, la prochaine problématique que nous allons traiter portera sur la découverte des dépendances multivaluées floues sachant qu'à ce jour, **aucun travail** n'existe dans la littérature sur ce sujet. Nous nous réservons donc cette thématique dans le cadre de la poursuite de nos travaux de recherche.

ANNEXE 1

Nous présentons ici quelques définitions et résultats relatifs aux ensembles ordonnés, aux treillis et plus particulièrement aux treillis de fermés. Pour plus de détails, le lecteur peut se référer aux ouvrages **[BIR 67][DAV 94].**

La notion de treillis est définie à partir des notions d'ensemble ordonné et de relation de couverture. Les définitions respectives de ces deux dernières notions sont données ci-après.

A.1.1- Ensemble ordonné et relation de couverture

Définition 1 : (Ensemble ordonné)

Soit E un ensemble. Un ordre ou ordre partiel sur l'ensemble E est une relation binaire $\leq$ sur les éléments de E tel que pour tout $x, y, z \in E$ nous avons les propriétés suivantes :

- Réflexivité : $x \leq x$,
- Anti-symétrie : $x \leq y$ et $y \leq x \Rightarrow x = y$,
- Transitivité : $x \leq y$ et $y \leq z \Rightarrow x \leq z$.

Un ensemble E doté d'une relation binaire d'ordre $\leq$, noté $(E, \leq)$, est appelé *ensemble ordonné*.

Remarque : les ensembles ordonnés considérés dans ce mémoire sont *finis.*

Définition 2 : (Relation de couverture)

Soit $(E, \leq)$ un ensemble ordonné et x, y deux éléments de E. La relation de couverture entre les éléments de E, notée $<\cdot$, obtenue en supprimant les couples de transitivité de $\leq$. La relation de couverture est définie par : $x <\cdot y$ si et seulement si $\forall\ z \in E,\ x \leq z \leq y \Rightarrow z = y$. On peut représenter un ensemble ordonné par son *graphe de couverture* (ou *diagramme de Hasse*) qui a pour sommets les éléments de E et pour arcs, les éléments de $<\cdot$.

Nous disons que *y couvre x* ou bien que *y* est un *successeur immédiat* de *x*.

Exemple 1 :

La figure **Fig.A1.1** est le diagramme de Hasse de l'ensemble ordonné { $1 <\cdot 2$, $1 <\cdot 3$, $2 <\cdot 6$ $3 <\cdot 4$, $3 <\cdot 5$, $4 <\cdot 6$, $5 <\cdot 6$}.

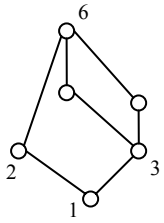


Fig.A1.1. Exemple de diagramme de Hasse d'un ensemble ordonné.

Nous allons maintenant exhiber quelques éléments particuliers d'un ensemble ordonné. Ces définitions sont utilisées par la suite lors de l'introduction de la notion de treillis.

Définition 3 : (l'élément maximal et **le plus grand élément d'un ensemble)**

Soient E un ensemble ordonné et $X \subseteq E$.

1. $x \in X$ est un *élément maximal* de X, si $x \leq y \in X \Rightarrow x = y$.
2. $x \in X$ est *le plus grand élément* de X, si $\forall y \in X, y \leq x$.

L'*élément minimal* et *le plus petit élément* de X sont définis dualement.

Remarque : si X a un plus grand élément (resp. Plus petit élément), il a exactement un élément maximal (resp. minimal). De plus, le plus grand élément de X (resp. plus petit élément), s'il existe, est unique.

Définition 4 : (élément top et élément bottom)

Soit E un ensemble ordonné. Le plus grand élément (resp. plus petit élément) de E, s'il existe, est appelé l'élément *top* (resp. bottom) de E, et noté par T (resp. $\perp$).

Définition 5 : (Majorant et minorant)

Soient E un ensemble ordonné et $X \subseteq E$. Un élément $x \in E$ est un *majorant* (resp. *minorant*) de X si $\forall\, y \in X, y \leq x$ (resp. $\forall y \in X, x \leq y$).

Définition 6 : (borne supérieure et borne inférieure)

Soient E un ensemble ordonné et $X \subseteq E$. Un élément x est la *borne supérieure,* appelée aussi *supremum* ou *join* de X, si x est le plus petit des majorants, et notée par *join*(X).

On définit de façon duale la *borne inférieure* (appelée aussi *infimum* ou *meet*) de X est le plus grand des minorants de X, et notée par *meet*(X).

Remarque : Si $X = \{x, y\}$, on note la borne supérieure (resp. borne inférieure) *join*($\{x, y\}$), (resp. *meet*($\{x, y\}$).

Exemple 2 :

Reprenons l'ensemble ordonné de la figure **Fig.A1.1** et considérons le sous-ensemble $\{3, 4, 5\}$.

Les éléments maximaux de $\{3, 4, 5\}$ sont 4 et 5. Le seul élément minimal est 3.

L'ensemble $\{3, 4, 5\}$ n'a pas de plus grand élément mais 3 est son plus petit élément.

L'ensemble des majorants de $\{3, 4, 5\}$ est $\{6\}$ et l'ensemble de ses minorants est $\{1, 3\}$.

La borne supérieure de $\{3, 4, 5\}$ est 6 et la borne inférieure est 3.

L'ensemble ordonné admet un plus grand élément ($T = 6$) et un plus petit élément ($\perp = 1$).

Les définitions que nous venons de présenter nous permet d'introduire à présent la notion de treillis.

Définition 7 : (Treillis et treillis complet)

Un ensemble ordonné $(E, \leq)$ non vide est un *treillis* si pour tout couple d'éléments $x, y \in E$ l'ensemble $\{x, y\}$ possèdent un plus petit majorant noté *Join*($\{x, y\}$) et un plus grand minorant noté *Meet*($\{x, y\}$).

Définition 8 : (Treillis complet)

L'ensemble ordonné $(E, \leq)$ est un *treillis complet* si pour tout sous ensemble $X \subseteq E$ les éléments *Join*(X) et *Meet*(X) existent.

Exemple 3 : l'ensemble ordonné de la figure Fig.A1.1 est un *treillis complet.*

Définition 9 : (sup-irréductible et inf-irréductible)

Soit E un treillis. Un élément $x \in E$ est un *sup-irréductible* (*join-irreductible*) si :

1. $x \neq \bot$,
2. $\forall\, y, z \in E, x = join(\{y, z\}) \Rightarrow (x = y)$ ou $(x = z)$.

Un élément *inf-irréductible* (*meet-irreductible*) est défini dualement. Nous notons $\mathcal{J}(E)$ (resp. $\mathcal{M}(E)$) l'ensemble des sup-irréductibles (resp. inf-irréductibles) du treillis E.

Remarque : les sup-irréductibles (resp. inf-irréductibles) sont les sommets du treillis possédant exactement un arc entrant (resp. sortant).

Exemple 4 :

En reprenant l'exemple 1, les sup-irréductibles sont {2, 3, 4, 5} et les inf-irréductibles sont {2, 4, 5}.

Lemme 1 :

Soit E un treillis.

1. $\forall A \in E, A = join(\{X \in \mathcal{J}(E) \mid X \leq A\})$,
2. $\forall A \in E, A = meet(\{X \in \mathcal{J}(E) \mid X \geq A\})$.

Le lemme ci-dessus indique qu'il est possible de retrouver tous les éléments du treillis à partir des irréductibles (inf-irréductibles et sup-irréductibles).

Exemple 5 :

En reprenant les inf-irréductibles de l'exemple 4 représentés par l'ensemble {2, 4, 5}, on peyt retrouver l'élément 3 = $join(\{4, 5\})$, l'élément 1 = $join(\{2, 4, 5\})$ et l'élément 6 = $join(\varnothing)$[4].

Nous allons maintenant présenter la notion de fermeture et montrer le lien qui existe entre fermeture et treillis.

[4] Par convention, T = $meet(\varnothing)$ et $\bot = join(\varnothing)$.

A.1.2- Treillis et fermeture

Définition 10 : (fermeture)

Soit E un ensemble. Une application $^+ : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ est un opérateur de *fermeture* (ou *fermeture*) sur E si, pour tout $X, Y \subseteq E$,

1. $X \subseteq X^+$ (extensivité),
2. si $X \subseteq Y$, alors $X^+ \subseteq Y^+$ (isotonie),
3. $X^{++} = X^+$ (idempotence).

Un sous-ensemble X de E est dit *fermé* (par rapport à $^+$) si $X^+ = X$.

Un fermé par $^+$ est donc un point fixe de $^+$, et comme $^+$ est idempotente, l'ensemble des fermés de $^+$est l'ensemble image de $^+$ de $\mathcal{P}(E)$.

Exemple 6 :

L'exemple le plus simple de fermeture est l'application identité : Id : $\mathcal{P}(E) \rightarrow \mathcal{P}(E)$, Id($X$) = X. l'ensemble des fermés pour cet opérateur est l'ensemble des $2^{|E|}$ sous-ensembles de E.

Théorème 1 :

Soit $^+$ un *opérateur de fermeture sur un ensemble E.* La *famille* $\{X_i \subseteq E \mid X_i^+ = X_i\}$ des *sous-ensembles fermés* de E *ordonnée par inclusion* forme *un treillis complet* dans lequel $join(\{X_i \mid i \in I\}) = (\cup_{i \in I} X_i)^+$ et $meet(\{X_i \mid i \in I\}) = (\cap_{i \in I} X_i)^+$. □

Ce théorème montre qu'à chaque opérateur de fermeture, nous pouvons associer un treillis complet dit *treillis de fermés.*

Exemple 7 :

La figure **Fig.A1.2** représente le treillis de fermés de l'application identité pour l'ensemble $\boldsymbol{R} = \{A, B, C, D, E\}$. Dans ce cas le treillis des fermés est le treillis des sous ensembles de $\boldsymbol{R}$.

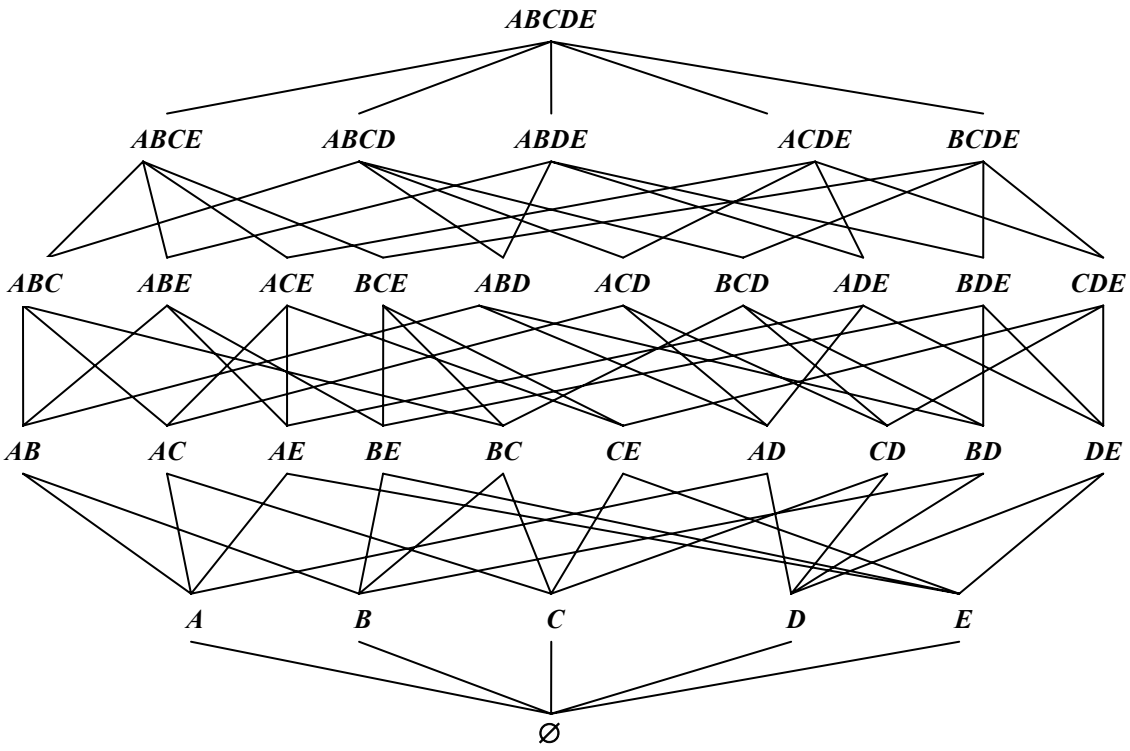


Fig.A1.2. Treillis des fermés des sous-ensembles de ***R***.

Définition 1 : (Fermeture d'un ensemble d'α-DFFs)

Soit ***R*** un schéma de relation et F un ensemble d'α-DFFs sur ***R***. F est un ensemble d'α-DFFs de la forme $X\sim>_\alpha Y$, tel que $X, Y \subseteq \boldsymbol{R}$. La fermeture de F, notée F^+ est l'ensemble de toutes les α-DFFs dérivable à partir de F. c'est-à-dire $X\sim>_\alpha Y \in F^+$ si $X\sim>_\alpha Y$ est valide dans toute relation floue où toutes les α-DFFs de F sont valides.

Soit $\tilde{r}$ une relation floue sur un schéma de relation. Si l'α-DFFs $X\sim>_\alpha Y$ est valide dans $\tilde{r}$, on écrit $\tilde{r} \models X\sim>_\alpha Y$. Similairement, si toute α-DFF de F est valide dans $\tilde{r}$, on écrit $\tilde{r} \models F$.

Soit F un ensemble d'α-DFFs, f une α-DFF et I l'ensemble des règles d'inférence. La preuve de f à partir de F en utilisant les règles d'inférence de I est une séquence d'α-DFFs f_1, $f_2,\ldots,f_n = f$ $(n \geq 1)$ tel que pour tout $i \in [1, n]$, soit :

- **(a)** $f_i \in F$, ou bien
- **(b)** f_i est la conséquence d'une règle d'inférence ρ dans I tel que l'α-DFFs dans la prémisse de ρ appartiennent à l'ensemble $\{f_j : 1 \leq j \leq i\}$.

Une α-DFF est prouvable à partir de F en utilisant les règles d'inférence de I, écrite $F \vdash^I f$, si il existe une preuve de f à partir de F utilisant I.

I est dit <u>correct</u> pour une implication logique d'α-DFF $F \vdash^I f$ implique $f \in F^+$.

I est <u>complet</u> pour une implication logique d'α-DFF si $f \in F^+$ implique $F \vdash^I f$

Définition 2 : (fermeture d'un ensemble d'attributs *X*)

Soit F un ensemble d'attributs défini sur le schéma de relation ***R***. la fermeture d'un ensemble d'attributs $X \subseteq \boldsymbol{R}$ par rapport à F, notée es, est définiX_F^+ar :

$$X_F^+ = \{A : X\sim>_\alpha A \in F^+\}.$$

A présent nous pouvons prouver le lemme suivant :

Lemme 1 :

Soit F un ensemble d'α-DFFs et $X \sim>_\alpha Y$ une α-DFF. Alors : $X \sim>_\alpha Y \in F^+ \Leftrightarrow Y \subseteq X_F^+$.

Preuve :

Soit $Y \subseteq X^+_F$. Alor $F \models^I X \sim>_\alpha A$, pour tout $A \in Y$. Donc, par la règle d'union FF5, cela donne $F \models^I X \sim>_\alpha Y$. Par conséquent, $X \sim>_\alpha Y \in F^+$. Inversement, soit $X \sim>_\alpha Y \in F^+$. Alors $F \models^I X \sim>_\alpha Y$. Par la règle de décomposition, $F \models^I X \sim>_\alpha A$ pour tout $A \in Y$. D'où, $Y \subseteq X^+_F$ □

Puisque les règles d'inférence pour les α-DFFs sont exactement les même que les dépendances fonctionnelles, X^+_F peut être calculé par l'algorithme suivant qui est le même que dans le cas classique **[ABI 95]**.

L'algorithme 1 calcule pour un ensemble F d'α-DFFs et un ensemble d'attributs X, la fermeture X^+_F de X par rapport à F.

Algorithme 1 : calcul de la fermeture d'un ensemble d'attributs

Entrée : F, un ensemble d'α-DFFs. X, un ensemble d'attributs
Sortie : X^+_F , la fermeture de X par rapport à F.

1 : *unused* := F ;
2 : *closure* := X ;
3 : **repeat**
4 : **if** $W \sim>_\alpha Z \in$ *unused* and $W \subseteq$ *closure* **then**
5 : *unused* := *unused* – { $W \sim>_\alpha Z$};
6 : *closure* := *closure* $\cup$ Z ;
7 : **until** no further changes ;
8 : output *closure*.

Théorème 1 :

L'ensemble I = {FF1, FF2, FF3, FF4, FF5, FF6} forme un système correct et complet pour l'inférence des α-DFFs. □

A.2.1- Preuve de correction :

I est dit correct pour une implication logique d'α-DFF $F \models^I f$ implique $f \in F^+$.

Supposons que $f \equiv W \sim>_\alpha V$ est dérivable par les règles d'inférence de l'ensemble I. $V = \{A_1, A_2, \ldots, A_k\}$. Alors pour chaque i, $W \sim>_\alpha A_i$ est valide (par la règle de décomposition). D'où, $V \subseteq W^+$.

Suppose, $V \subseteq W^+$, selon la définition 2, pour tout i, $W \sim\!>_\alpha A_i$ est dérivée par les règles d'inférence de I (règle de réflexivité). Alors en utilisant la règle d'Union, on obtient $W \sim\!>_\alpha V$ est valide.

A.2.2- Preuve de complétude :

On montre que : $F \models X \sim\!>_\alpha Y$ implique $F \vdash X \sim\!>_\alpha Y$. Premièrement, on montre : par induction basée sur l'algorithme 1 que $F \vdash X \sim\!> X_F^+$ est vrai. Soit $closure_i$ la valeur de *closure* après $i^{ème}$ itération de la même exécution de l'algorithme 1 sur un ensemble d'α-DDFs. On obtient $closure_i = X$. De manière inductive, supposons une prouve $\sigma_1, \sigma_2, \ldots, \sigma_{ki}$ de $X \sim\!>_\alpha closure_i$ est construite. Le cas où $i = 0$ est obtenu par FF1. Supposons que $W \sim\!>_\alpha V$ est utilisée à l'(i+1)-itération. Ce qui donne $W \sim\!>_\alpha closure_i$ et $closure_{i+1} = closure_i \cup V$. Les étapes suivantes expliquent la preuve.

$\sigma_{ki+1} = W \sim\!>_\alpha V$	par F
$\sigma_{ki+2} = closure_i \sim\!>_\alpha W$	par FF1
$\sigma_{ki+3} = closure_i \sim\!>_\alpha V$	par FF3
$\sigma_{ki+4} = closure_i \sim\!>_\alpha closure_{i+1}$	par FF2
$\sigma_{ki+5} = X \sim\!>_\alpha closure_{i+1}$	par FF3

La fin de cette construction, on aura la preuve $\sigma_1, \sigma_2, \ldots, \sigma_n$ de $X \sim\!>_\alpha X_F^+$. Par le lemme1, $Y \subseteq X_F^+$ D'où, par la règle FF1, $X_F^+ \sim\!>_\alpha Y$ et par la règle FF3 $X \sim\!>_\alpha Y$. CQFD

BIBLIOGRAPHIE

[ABI 95] S. Abiteboul, R. Hull, V. Vianu. "*Foundation of Databases*". In: Addition-Wesley Publication Company Inc., USA, 1995.

[AGR 93] R. Agrawal, T. Imielinski, A. Swami. "*Mining associations rules between sets of items in large databases*". In: Proc. of the ACM SIGMOD Conf. on Management of Data, Washington, D.C, pp. 207-216, 1993.

[AGR 94] R. Agrawal, R. Skirant. "*Fast Algorithm for mining association rules*". In: IBM Research Report RJ9839, IBM Almaden Research Center, San Jose, CA, June 1994.

[ARM 74] W.W. Armstrong. "*Dependency Structures of Databases Relationships*". In: IFIP World Congress, North-Holland Ed., pp. 580-583, 1974.

[BER 76] C. Berge. "*Graphs and hypergraphs*". In: North-Holland Mathematical Library 6, American Elservier, 2nd rev.ed. edition, 1976.

[BER 98] P.A. Bernstein, M.L. Brodie, S. Ceri, D..J. DeWitt, M.J. Franklin, H. Garacia-Molina, J. Gray, G. Held, J.M. Hellerstein, H.V. Jagadish, M. Lesk, D. Maier, J.F. Naughton, H. Pirahesh, M. Stonebraker, J.D. Ullman. "*The Asilomar report on database research*". In: ACM Sigmod Record, Vol.27, N°.04, pp.74-80, 1998.

[BHU 93] B. Bhuniya, P. Niyogi, "*Lossless join property in fuzzy relational databases*". In: Data Knowledge Eng. Vol. 11, issue 02, pp 109-124, 1993.

[BIA 01] J. Biais, Associé-Gérant – INDICTA. " *L'extraction de données : recherche du contenus, d'informations ou de connaissances* ? ". Dans : Conférence Knowledge Management, EFE, Paris, 4 juillet 2001.

[BIR 67] G. Birkhoff. "*Lattices theory*". In: American Mathematical Society Colloquium Publications, vol. 25, American Mathematical Society, 3rd edition, 1967.

[BOS 93] P. Bosc, D. Dubois, H. Prade. "*An introduction to fuzzy sets and possibility theory based approaches to the treatment of uncertainty and imprecision in database management systems*". In Proceedings of the 2nd Workshop on Uncertainty Management in Information Systems: From needs to solutions, Catalina, CA, USA , 1993.

[BOS 97b] P. Bosc, L. Liétard, O. Pivert. "*Functional Dependencies Revisited Under Graduality and Imprecision*". In: Fuzzy Information Processing Society. NAFIPS '97. Annual Meeting of the North American**,** pp. 57-62, 1997.

[BOS 04] P. Bosc, L. Liétard, O. Pivert, D. Rocacher. Livre de base de données "*Gradualité et imprécision dans les bases de données : Ensembles flous, requêtes flexibles et interrogation de données mal connues*". Dans : Ellipses Edition Marketing S.A., 2004.

[BUC 82] B.P. Buckles, F.E. Petry. "*A Fuzzy Representation of Data for Relational Databases*". In: Fuzzy Sets and Systems, Vol. 7, pp. 213-226, 1982.

[CHA 98] S. Chaudhuri, V.R. Narasayya. "*Autoadmin' what-if' index analysis utility*". In: L.M. Haas, Ashutosh Tiwary, editors. In: Proceeding of the ACM SIGMOD International Conference on Management of Data (SIG-MOD), Seattle, Washington, USA, pp. 367-378. ACM Press, June 1998.

[CHA 00] S. Chaudhuri, V.R. Narasayya. "*Automating statistics management for query optimizers*". In: Proceeding of the Sixteenth IEEE International Conference on Data Engineering (ICDE), San Diago, California, USA, pp. 339-348, IEEE Computer Society Press, 2000.

[CHE 95] G. Chen. "*Fuzzy functional dependency and a series of design issues of fuzy relational databases*". In: P. Bosc, J. Kacprzyk (Eds.), Studies in Fuzziness: Fuzziness in Database Management Systems, Physica-Verlag,Worzburg, pp. 166-185, 1995.

[CHE 95b] G. Chen, E.E. Kerre, J. Vandenbulcke. "*The dependency-preserving decomposition and a testing algorithm in a fuzzy relational data model* ". In: Fuzzy Sets and Systems, vol. 72, pp. 27–37, 1995.

[CIV 86] M.R. Civanlar and H.J. Trussel. "*Constructing membership functions using statistical data*". Fuzzy Sets and Systems, Vol. 18, pp. 1-13, 1986.

[COD 70] E.F. Codd. "*A relational model for large shared data banks*". In: Communications of the ACM, vol. 13, 1970.

[COS 86] S.S. Cosmadakis, P.C Kanellakis, N. Spyratos. "*Partition semantics for relations* ". In: Journal of Computer and System Sciences, Vol. 33, N°. 2, pp. 203-233, 1986.

[CUB 93] J.C. Cubero, J.M. Medina, M.A. Vila. "*Influence of granularity level in fuzzy functional dependencies*". In: Symbolic and Quantitative Approaches to Reasoning and Uncertainty (Proc. ECSQARU'93), Lecture Notes in Computer Science, Springer Verlag, Berlin, vol. 747, pp. 73-78, 1993.

[CUB 94] J.C. Cubero, M.A. Vila. "*A new definition of fuzzy functional dependency in fuzzy relational databases*". In: Internat. J. Intell. Systems, vol. 9, issue 5, pp. 441–448, 1994.

[DEL 87] M. Delgado and S. Moral. "*On the concept of possibility-probability consistency*". Fuzzy Sets and Systems, Vol. 21 pp. 311-318, 1987.

[DAV 94] B.A. Davey, H.A. Priestley. "*Introduction to Lattices and Order*". In: Cambridge university press, fourth edition, 1994.

[DJO 07] Y. Djouadi, S. Redaoui, K. Amroun. "*Mining Association Rules Under Imprecision and Vagueness : towards a possibilistic Approach*". In: FUZZ-IEEE'07, pp. 722-725, Londres, 23-27 juillet, 2007.

[DJO 07b] Y. Djouadi, S. Redaoui, K.Amroun. "*Gradual Uncertainty Association Rules Mining*". In: COSI'07, pp. 315-326, Oran, 11-13 juin, 2007.

[DUB 80] D. Dubois, H. Prade. "*Fuzzy sets and systems: theory and applications*". In: Academie Press, New York, 1980.

[DUB 86] D. Dubois, H. Prade. "*A set-theoretic view of belief functions: logical operations and approximations by fuzzy sets*". International Journal of General Systems, Vol. 12, pp. 193-226, 1986.

[DUB 88] D. Dubois, H. Prade. Livre "*Théorie des possibilities*: *Application à* représentation *des connaissances en informatique*". Edition Masson Paris Milan Barcelone Mexico, 1988.

[DUB 88b] D. Dubois, H. Prade. "*default reasoning and possibility theory*". In: Artificial Intelligence, vol. 35, pp. 243-257, 1988.

[DUB 91] D. Dubois, H. Prade, and S. Sandri. *On possibility/probability transformations*. In Proceedings of the Fourth Int. Fuzzy Systems Association World Congress (IFSA'91), Brussels, Belgium, pages 50–53, 1991.

[DUB 94] D. Dubois, H. Prade. "*Logique Floue*". Série Arago. Observatoire français des techniques avancées, 1994.

[DUB 00] D. Dubois, H. Prade. "*Fundamentals of fuzzy sets*". In: Kluwer Academic Publishers, 2000.

[DUB 00b] D. Dubois, H.T. Nguyen, H. Prade. "*Possibility theory, probability and fuzzy sets* : *Misunderstandings, bridges and gaps*". In D. Dubois and H. Prade, editors, Fundamentals of Fuzzy sets, pp. 343-438. Kluwer Academic Publishers, Boston, 2000.

[DUB 04] D. Dubois, L. Foulloy, G. Mauris, H. Prade. "*Probability-possibility transformations, triangular fuzzy sets and probabilistic inequalities*". Reliable Computing, 10 :273–297, 2004.

[FAY 91] U.M. Fayyad, G. Piatetsky-Shapiro, C. Matheus. "*Knowledge discovery and databases: an overview*". In: G. PIATETSKY-SHAPIRO, W. FRAWLEY, Eds., Knowledge discovery and databases. Menlo Park : AAAI Press, pp.1-27, 1991.

[FAY 96] U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth. "*From Data Mining to Knowledge Discovery: An Overview*". In: advances in Knowledge Discovery and Data Mining, pp. 1-34, 1996.

[FAY 96b] U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth. "*Knowledge discovery and data mining: Towards a unifying framework* ". In: Proc. Of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD'96). Portland, Oregon, pp. 82-88, August 1996.

[FRA 92] W. Frawley, G. Piatetsky-Shapiro, C. Matheus. "*Knowledge Discovery in Databases: An Overview*". In: AI Magazine, pp. 213-228. ISSN 0738-4602, 1992.

[GAR 03] G. Gardarin. Livre "*Bases de données*". Edition Eyrolles, 2003.

[GOT 90] G. Gottlob, L. Libkin. "*Investigations on Armstrong Relations, Dependency Inference, and Excluded Functional Dependencies*". *Acta* Cybernetica, Vol. 9, N°. 4, pp. 385-402, 1990.

[GRA 98] G. Graefe, U.M. Fayyad, S. Chaudhuri. "*On the efficient gathering of sufficient statistics for classification from large sql databases*". In: R. Agrawal, P.E. Stolorz, G. Piatetsky-Shapiro, editors, Proceedings of the Forth International Conference on Knowledge Discovery and Data Mining (KDD), New York City, New York, USA, pp. 204-208, AAAI/MIT Press, 1998.

[GUI 86] S. Guillaume. "*Traitement des données volumineuses : Mesure et algorithmes d'extraction de règles d'association et règles ordinales* ". *Thèse* de doctorat, Université de Nantes, 2000.

[HAT 96] K. Hätönen, M. Klemettinen, H. Mannila, P.Ronkainen, H. Toivonen. "*Knowledge discovery from telecommunication network alarm databases*". In: Proceedings of the 12th International Conference on Data Enginnering (ICDE'96), pp. 115-122. IEEE Computer Society Press, 1996.

[HUH 99] Y. Huhtala, J. Kärkkäinen, P. Porkka, H. Toivonen. "*TANE: An efficient algorithm for discovering functional and approximate dependencies*". In: The Computer Journal, vol. 42, issue. 02, pp 100-111, 1999.

[KIS 91] A. Kiss. "*λ-decomposition of fuzzy relational databases*". Annales Univ. Sci. Budapest., Sect. Comp., vol.12, pp.133-142, 1991.

[KIV 95] J. Kivinen and H.Mannila. "*Approximate Dependency Inference from Relations*". In: *Theoretical Computer Science*, 149(1) :129–149, 1995.

[KLE 97] M. Klemettinen, H. Mannila, H. Toivonen. "*A data-minig methodology and its application to semi-automatic knowledge acquisition*". In: Proceedings of the 8th International Conference on Databases and Expert systems Applications (DEXA'97), Lecture Notes in Computer Science, Vol. 1308, pp. 670-677. Springer-Verlag, 1997.

[LAS 00] V. Lasserre, G. Mauris, L. Foulloy. "*A simple possibilistic modelisation of measurement uncertainty*". In: L.A. Zadeh Eds. B. Bouchon-Meunier, R.R. Yager, editor, Uncertainty in Intelligent and Information Systems, World Scientific, pp. 58-69, 2000.

[LEV 99] M. Levene, G. Loizou. "*A Guided Tour of Relational Databases and Beyond*". In: Springer-verlag London Limited, 1999.

[LOP 00] S. Lopes. "*Data Mining: Algorithmes d'analyse de schémas de bases de données relationnelles*". Thèse de l'Université de Blaise Pascal, Clemont Ferrand II, 2000.

[LOP 00b] S. Lopes, J.M Petit, L. Lakhal. "*Dep-miner: Un algorithme d'extraction des dépendances fonctionnelles*". Dans : Technique et Science Informatique (TSI), vol. 19(10), pp. 1399-1428, 2000.

[MAN 94] H. Mannila, H. Toivonen, A.I. Verkamo. "*Efficient algorithms for discovering association rules*". In: AAAI'94 Workshop on Knowledge Discovery in Databases, pp. 181-192. AAAIPress, 1994.

[MAN 94b] H. Mannila, K.J Räihä. "*Algorithms for inferring functional dependencies from relations*". In: Data and Knowledge Engineering, vol. 12, issue. 01, pp. 83-99, 1994.

[MAN 94c] H. Mannila, K.J Räihä. "*The Design of Relational Databases*". Addison Wesley, 1994.

[MAS 06] M.H Masson, T. Denoeux. "*Inferring a possibility distribution from empirical data*". In: Fuzzy Sets and Systems, vol.157, issue. 03, pp. 319-340, 2006.

[MED 94] J.M. Medina, O. Pons, M.A. Vila. "*GEFRED : A Generalized Model for Fuzzy Relational Databases*". In: Information Sciences, Vol. 77, N°.6, pp. 87-109, 1994.

[MEG 04] N. Méger. "*Recherche automatique des fenêtres temporelles optimales des motifs séquentiels* ". Thèse présentée devant l'Institut National des Sciences Appliquées de Lyon, 2004.

[NOV 01] N. Novelli, R. Cicchetti. "*FUN : An Efficient Algorithm for Mining Functional and Embedded Dependencies*". In: Proceedings of the 8th biennal Conference on Database Theory (ICDT'01), Lecture Notes in Computer Science, pages 189–203, London, UK, January 2001.

[PRA 82] H. Prade, C. Testemale. "*Fuzzy Relational Databases : Representational issues and Reduction Using Similarity Measures*". Information Sciences, 38(2), pp. 118-126, 1987.

[PRA 84] H. Prade, C. Testemale. "*Generalizing database relational algebra for the treatment of incomplete or uncertain information and vague queries*". Information Sciences, vol. 34, pp. 115–143, 1984.

[PET 96] J.M. Petit, F. Toumani, J.F. Boulicaut, J. Kouloumdjian. *Towards the Reverse Engineering of Denormalized Relational Databases*. In *Proceedings of the 12th International Conference on Data Engineering (ICDE'96)*, IEEE Computer Society, pages 218–227, New Orleans, US, February 1996.

[PON 95] O. Pons, "*Representacion Logica de Bases de Datos Difusas Fundamentos Teoricos e Implementacion*". Thèse de doctorat 1995.

[RAJ 86] K.V.S.V.N. Raju, A.K. Mazumdar. "*Fuzzy dependencies in fuzzy relations*". In: Proc. 2nd Internat. Conf. on Data Engineering, Los Angeles, pp. 312-319, February 1986.

[RAJ 88] K.V.S.V.N. Raju, A.K. Mazumdar. "*Fuzzy dependencies and lossless join decomposition of fuzzy relational database systems*". In: ACM Transaction of Database Systems, vol. 13, issue. 02, pp. 129-166, 1988.

[RED 06] S. Redaoui, Y. Djouadi. "*Découverte des Règles d'Association : Application aux Données Imprécises*". Dans : Logique Floue et ses applications, Toulouse, France, pp. 195-202, 19-20 Octobre, 2006.

[SAF 76] G. Shafer. "*A mathematical theory of evidence*". In: Princeton University Press, 1976.

[SAV 95] A. Savasere, E. Omiecinski, S. Navathe. "*An efficient algorithm for mining association rules in large databases*". In: Proceeding of the 21st International conference on Very Large Data Bases (VLDB'95), pp. 432-444, Morgan Kaufmann, September 1995.

[SRI 96] R. Srikant. "*Fast algorithms for minig association rules and sequential patterns*". PHD thesis, University of Wisconsin, 1996.

[THA 95] B. Thalheim, L. Libkin, editors. "*Semantics in Databases*", Vol. 1358, Springer Verlag, lecture notes in computer science edition, 1995.

[TOI 96b] H. Toivonen**.** "*Sampling large databases for association rules*". In: Proceeding of the 22nd international conference on Very Large Data Bases (VLDB'96), pages. 134-145. Morgan Kaufmann, September 1996.

[UMA 80] M. Umano, S. Fukami, M. Mizumoto, K. Tanaka. "*Retrieval Processing from Fuzzy Databases*". In; Technical Reports of IECE of Japan, Vol. 80, N°. 204 (on Automata and Languages), pp. 45-54, AL80-50, 1980.

[WAN 00] S.L. Wang, J.S. Tsai, T.P. Hong. "*Mining functional dependencies from fuzzy relational databases*". In: Proceedings of the 2000 ACM symposium on Applied computing, Vol. 01, pp. 490-493, 2000.

[WYS 01] C. Wyss, C. Giannella, and E. Robertson. FastFDs : A Heuristic-Driven, Depth-First Algorithm for Mining Functional Dependencies from Relation Instances. In *Proceedings of Data Warehousing and Knowledge Discovery (DaWaK 2001)*, Munich, Germany, 2001.

[YAG 80] R.R. Yager. "*An approach to inference in approximate reasoning*". In: Journal of Man-Machine Studies, vol. 19, pp. 195-227, 1980.

[ZAD 65] L.A. Zadeh. "*Fuzzy Sets*". In: Information and Control, Vol. 08, pp. 338-352, 1965.

[ZAD 78] L.A. Zadeh. "*Fuzzy Sets as a basis for a theory of possibility*". In: Fuzzy Sets and Systems, vol. 01, pp 3-28, 1978.

[ZAN 81] C. Zaniolo, M.A. Melkanoff. "*On the design of relational Database Schemata*". In: ACM Transactions on Database Systems, Vol. 06, n°. 01, pp. 1-47, 1981.

[ZIG 01] D.A. Zighed, Y. Kodratoff, A. Napoli. "*Extraction de connaissance à partir d'une base de données*". Dans : Bulletin LFIA'01, 2001.