UNIVERSITÉ DE LORRAINE

AVERTISSEMENT

Ce document est le fruit d'un long travail approuvé par le jury de soutenance et mis à disposition de l'ensemble de la communauté universitaire élargie.

Il est soumis à la propriété intellectuelle de l'auteur. Ceci implique une obligation de citation et de référencement lors de l'utilisation de ce document.

D'autre part, toute contrefaçon, plagiat, reproduction illicite encourt une poursuite pénale.

Contact : ddoc-theses-contact@univ-lorraine.fr

LIENS

Code de la Propriété Intellectuelle. articles L 122. 4
Code de la Propriété Intellectuelle. articles L 335.2- L 335.10
http://www.cfcopies.com/V2/leg/leg_droi.php
http://www.culture.gouv.fr/culture/infos-pratiques/droits/protection.htm

Université du Québec À Montréal

École Doctorale IAEM Lorraine

ANALYSE RELATIONNELLE DE CONCEPTS : UNE MÉTHODE POLYVALENTE POUR L'EXTRACTION DE CONNAISSANCES

THÈSE EN COTUTELLE

Présentée et soutenue publiquement le 09 novembre 2020

Pour l'obtention des titres de

Doctorat de l'Université du Québec À Montréal
En Informatique

Doctorat de l'Université de Lorraine
En Automatique, Traitement du signal et des Images, et Génie Informatique

par

Mickaël WAJNBERG

Composition du jury

Président :
Prof. Abdoulaye Baniré Diallo, Professeur à l'Université du Québec à Montréal, Canada

Rapporteurs :
Prof. Marianne Huchard Professeur à Université Montpellier II, France
Dr. Virginie Goepp, Maître de Conférences, HdR à l'INSA de Université de Strasbourg

Examinateurs :
Prof. Alexandre Blondin-Massé, Professeur à l'Université du Québec à Montréal, Canada *(Directeur de thèse)*
Prof. Petko Valtchev, Professeur à l'Université du Québec à Montréal, Canada *(Co-directeur de thèse)*
Prof. Hervé Panetto, Professeur à l'Université de Lorraine, France *(Directeur de thèse)*
Dr. Mario Lezoche, Maître de Conférences à l'Université de Lorraine, France *(Co-directeur de thèse)*
Prof. Hind Bril-El Haouzi, Professeur à l'Université de Lorraine, France

UNIVERSITÉ DU QUÉBEC À MONTRÉAL
UNIVERSITÉ DE LORRAINE

ANALYSE RELATIONNELLE DE CONCEPTS : UNE MÉTHODE POLYVALENTE POUR L'EXTRACTION DE CONNAISSANCE

THÈSE
PRÉSENTÉE
COMME EXIGENCE PARTIELLE
DU DOCTORAT EN INFORMATIQUE

PAR
MICKAEL WAJNBERG

NOVEMBRE 2020

REMERCIEMENTS

Entre ce qui marche, ce qui ne marche pas et ce dont on n'est pas trop sûr, la thèse de doctorat a l'effet de montagnes russes sur le moral. Cette tranche de vie, passionnante, mais pour le moins intense, a été illuminée grâce à la présence de certaines personnes que je tiens à remercier ici !

Tout d'abord, pour leur exploit de m'avoir supporté toutes ces années, je tiens à remercier du fond du coeur mes superviseurs. Notamment, Petko qui m'a fait découvrir un monde passionnant, prêt à rester jusqu'à 5 heures du matin pour travailler avec moi, Mario qui m'a énormément apporté sur le plan personnel (et gastronomique) depuis nos débuts à ScuolaNonScuola et enfin Alexandre, qui m'a pris sous son aile depuis la maîtrise et qui m'a donné goût à la recherche, merci pour ton soutien positif, ton énergie et d'avoir été autant présent !

À mes parents, qui m'ont toujours poussé à me réaliser à mon plein potentiel et qui sont toujours là pour tout et (surtout) n'importe quoi, grâce à leur «barre de faire», et les parties d'échecs. À mon aventurière de soeurette qui était là tout le temps pour me ramener les pieds sur terre, et pour m'expliquer les formules de «la variable presque dans la boîte». À ma grand-mère Mana pour son amour et son soutien absolument toutes catégories durant toute ma vie, notamment cette thèse. Évidemment Tatie, Tonton, Sarah, Jo', Mamie et Papi qui ont joué un rôle majeur dans le fait que je sois qui je suis aujourd'hui.

Un grand merci aussi à tous mes amis du labo sans qui ça n'aurait pas été pareil. À Marie-Claude Côté, toujours là pour me sauver quand je me perdais dans les dédales administratifs. Et à tous mes amis d'enfance, qui m'ont soutenu tout au

long de l'aventure (Greg, sans toi, cette thèse aurait des figures qui bavent) ! Coloc, Vincent, Aurélie, Fabien qui ont maintenu ma santé mentale à flot ! Sylvain pour nos chantouillages. Louis pour m'avoir motivé à avancer. Élisa qui m'a sincèrement transformé, et sans qui je n'aurais pas tenté l'aventure cotutelle. Caroline qui a su m'apporter un véritable soutien dans les moments difficiles. Merci à vous tous !

TABLE DES MATIÈRES

LISTE DES TABLEAUX viii
LISTE DES FIGURES x
RÉSUMÉ xiii
INTRODUCTION 1

I Théorie **6**

CHAPITRE I EXTRACTION DE CONNAISSANCE 7
1.1 Définition de la connaissance 9
1.2 Représentation de la connaissance 13
1.2.1 Formalisation 13
1.2.2 Représentation de la connaissance 16
1.3 Extraction de connaissance 17
1.3.1 Fouille de données multi-relationnelles 18
1.3.2 Extraction d'ontologie 20
1.3.3 Extraction des règles d'association 21

CHAPITRE II ANALYSE FORMELLE DE CONCEPTS 30
2.1 Treillis 30
2.1.1 Ensemble partiellement ordonné 30
2.1.2 Treillis complet 33
2.2 Analyse formelle de concepts 34
2.3 Règles d'association 44
2.3.1 Représentation 45
2.3.2 Base de représentation des règles produite par l'AFC 48

CHAPITRE III ANALYSE RELATIONNELLE DE CONCEPTS 52
3.1 Extensions relationnelles de l'AFC 52
3.2 Formalisme de l'analyse relationnelle de concepts 56
3.2.1 Famille relationnelle de contextes 57
3.2.2 Opérateur de propositionnalisation 60
3.2.3 Graduation 61
3.2.4 Cadre algorithmique 63
3.3 Interprétation des treillis relationnels 66

CHAPITRE IV FONDEMENTS, LIMITES ET EXTENSIONS DE L'ANALYSE RELATIONNELLE DE CONCEPTS 73
4.1 Concept formel pour l'ARC 73
4.1.1 Concept formel évolutif 74
4.1.2 Concept formel strict 75
4.1.3 Accumulation 78
4.2 Améliorations de l'ARC 78
4.2.1 Règles d'association 79
4.2.2 Raisonnement 88
4.3 Algorithme 90
4.4 Comparaison avec le paradigme non-relationnel 92
4.4.1 Opération de semi-jointure en cas mono-relationel 94
4.4.2 Opération d'agrégation en cas mono-relationel 102
4.4.3 Généralisation à une famille multi-relationnelle 107

II Applications 111

CHAPITRE V AIDE À LA PRISE DE DÉCISION INDUSTRIELLE . . 112
5.1 Introduction 112
5.2 Cadre d'étude 113

5.3 Modélisation du problème de maintenance 114
5.3.1 Données 115
5.4 Expérience 1 : Restriction aux pièces problématiques 117
5.4.1 Conception de la famille relationnelle 117
5.4.2 Protocole 120
5.4.3 Résultats 120
5.5 Expérience 2 : Sans contrainte 123
5.5.1 Conception de la famille relationnelle 123
5.5.2 Résultats 124

CHAPITRE VI ANCRAGE SYMBOLIQUE ET LEXIQUE MENTAL . 130
6.1 Contexte psycholinguistique 131
6.2 Formalisation 133
6.2.1 Lexiques 134
6.2.2 Graphes 135
6.3 Le *jeu du dictionnaire* 137
6.4 Analyse relationnelle 140
6.4.1 Données 140
6.4.2 Modélisation 143
6.4.3 Résultats 145

CHAPITRE VII INGÉNIERIE DE CONNAISSANCES PAR ANALYSE DE CONCEPTS 150
7.1 Introduction 150
7.2 Restructuration 152
7.3 Détection des anomalies dans le typage des données 160
7.4 Raffinement du schéma 164

CONCLUSION 172

RÉFÉRENCES 176

LISTE DES TABLEAUX

Tableau Page

2.1 Contexte formel de l'exemple 6 36

3.1 Contexte K_D. Décrit un ensemble de médicaments au travers d'attributs représentant leur molécule active et leurs effets secondaires connus 58

3.2 Contexte K_P. Décrit un ensemble de patients au travers d'attributs d'état civil et de symptômes médicaux présents 58

3.3 Relation takes 59

3.4 Relation takes^{-1} 59

3.5 Différents opérateurs de propositionnalisation 62

3.6 Contexte $K_{\mathcal{P}}^{+}$ 64

4.1 Contexte des voitures 96

4.2 Contexte des personnes 96

4.3 Relation pos 96

4.4 Jointure de la famille relationnelle de l'exemple 28 96

4.5 Agrégation de la famille relationnelle de l'exemple 30 103

4.6 Agrégation de la famille relationnelle de l'exemple 32 107

5.1 Attributs de la famille relationnelle de la première expérimentation 118

5.2 Famille relationnelle de l'exemple 33 119

5.3 Détail des attributs $a_{i;j;k}$ 124

5.4 Extrait des règles produites par ARC 125

6.1 Exemple de lexique complet non-ambigu. 135

6.2 Caractéristiques des graphes relatifs aux dictionnaires 141

6.3 Extrait de la table de propriétés psycholinguistiques. 142

6.4 Nombre de dictionnaires par mot racine 142

6.5 Discrétisation des variables structurelles et psycholinguistiques . . 143

6.6 Extrait des règles d'associations produites de la RCF figure 6.4 . . 145

7.1 RCF correspondant à la figure 7.1. Les associations sont représentées par des paires $V-W$, où V est l'initiale de l'ensemble domaine, et W du codomaine de l'association. Par exemple M-G représente l'association entre Musicien et Guitare. *e-Guit.* est l'abbréviation pour la classe *Guitare électrique*. 156

7.2 Ensemble des contextes formels que l'on peut générer à partir des données 162

7.3 Table des relations 163

7.4 Contexte diagonalisé pour les organisations *Beatles*, *Beck*, *Democratic* ayant les attributs *Grammy* précisant si une organisation a obtenue le prix de musique *Grammy award* de l'album de l'année et *centenaire* précise si une organisation a plus de 100 ans 166

7.5 Extrait du tableau de définition récursive des attributs relationnels par concept 168

LISTE DES FIGURES

Figure Page

1.1 Pyramide DIKW (longlivetheux, 2005) 10

2.1 Exemple de *poset* $(X, <)$ représenté sous forme de graphe orienté. 32

2.2 Diagramme de Hasse du *poset* de la figure 2.1 33

2.3 Treillis complet des partitions de l'ensemble $\{1, 2, 3, 4\}$ 35

2.4 Représentation en graphe biparti du contexte K donné au tableau 2.1 38

2.5 Diagrammes de Hasse du treillis de concepts du contexte 2.1 . . . 41

3.1 Treillis de l'exemple 14 59

3.2 Treillis $\mathcal{P}$: Extension 1 64

3.3 Extrait du multigraphe de la famille relationnelle de l'exemple 18 (Nica *et al.*, 2016a) 69

3.4 SCPO du concept `CKVT_6` de la figure 3.3 (Nica *et al.*, 2016a) . . 69

3.5 Hiérarchie des SCPO (Nica *et al.*, 2016a) 70

3.6 Composantes fortement connexes du multigraphe des treillis (Ferré et Cellier, 2018) 71

4.1 Treillis $\mathcal{P}$: point-fixe 82

4.2 Treillis $\mathcal{D}$: point-fixe 82

4.3 Extrait du multigraphe au point fixe de la FRC de l'exemple 25 . 83

4.4 Extrait du multigraphe de création de l'exemple 25 83

4.5 Le multigraphe $\mathcal{G}_{n+1}$ est composé des concepts existant à l'itération $n+1$ et, pour chacun de ces concepts, des attributs relationnels présents dans l'intension à l'itération de leur création. Il est composé du sous-multigraphe $\mathcal{G}_n$, son homologue à l'itération n, ainsi que des concepts générés à l'itération $n+1$ représentés par les cercles dans le bandeau $\mathcal{G}_{n+1} \setminus \mathcal{G}_n$. Les arcs représentent les attributs relationnels. Cette figure illustre le fait qu'il n'y a pas d'arcs entre ces nouveaux concepts et qu'il n'y a pas d'arc sortant de $\mathcal{G}_n$. 84

4.6 Treillis figure 4.2 90

4.7 Modélisation ARC, deux tables K_1,K_2 et une relation r 95

4.8 Modélisation après semi-jointure de la famille figure 4.7 95

4.9 Modélisation après agrégation de la famille figure 4.7 102

4.10 Modélisation jointure mutiples relations 109

4.11 Modélisation jointure mutiples relations par duplication 109

5.1 Triangle des contraintes de production. 114

5.2 Schéma de la machine, avec annotation des étapes de production . 115

5.3 Processus d'analyse des données 116

5.4 Schéma traduisant la situation de la règle #4. Les concepts PC indiquent les concepts des pièces, PB ceux des problèmes. Les flèches pleines marquent la relation d'ordre dans les treillis, les pointillés les reférences par attribut relationnel. PB N+1 porte exclusivement un attribut *relationnel* et donc ne peut être un concept-objet : il regroupe donc plusieurs problèmes. Ces problèmes co-occurrent avec des pièces de PC2. 129

6.1 Le graphe orienté du lexique complet non-ambigu de l'exemple 34. 137

6.2 Capture d'écran du jeu des dictionnaires (Vincent-Lamarre *et al.*, 2017). La zone 1 présente un menu déroulant où le joueur peut sélectionner un mot et produire sa définition dans la zone 2. La zone 3 montre toutes les définitions produites par le joueur. Les mots écrits en vert sont définis alors que ceux en noir non. Les mots en gris sont les lexèmes fonctionnels. La zone 4 présente la progression du joueur. 139

6.3 Graphe représentant un dictionnaire de cinq mots : *being*, *humain*, *live*, *living* et *person* 140

6.4 Schéma de la RCF du *jeu du dictionnaire* 144

7.1 Diagramme de classes de l'exemple de restructuration 153

7.2 Treillis au point fixe de la RCF du tableau 7.1. Cls représente les concepts du treillis des classes, Prop (pour *property*) dénote les concepts des associations. 155

7.3 Treillis des associations, au point fixe. Les attributs d'un noeud représentent complètement l'intension du concept décrit : les intensions ne sont pas réduite pas la compression «non relationnelle» appliquée aux diagrammes de Hasse. 158

7.4 Schéma de classes après restructuration 160

7.5 Diagramme de classe extrait de *DBpedia*. 161

7.6 Treillis au point fixe des contextes K_{Pers} et K_{Org} en utilisant la compression relationnelle et l'explicitation des attributs relationnels avec les générateurs canoniques. 169

7.7 Treillis au point fixe du contexte K obtenu avec l'ARC 170

7.8 Treillis au point fixe du contexte K après optimisations 171

7.9 Schéma raffiné du contexte K 171

RÉSUMÉ

À une époque où les données, souvent interprétées comme une «réalité terrain», sont produites dans des quantités gargantuesques, un besoin de compréhension et d'interprétation de ces données se développe en parallèle. Les jeux de données étant maintenant principalement relationnels, il convient de développer des méthodes qui permettent d'extraire de l'information pertinente décrivant à la fois les objets et les relations entre eux. Les règles d'association, adjointes des mesures de confiance et de support, décrivent les co-occurences entre les caractéristiques des objets et permettent d'exprimer et d'évaluer de manière explicite l'information contenue dans un jeu de données. Dans cette thèse on présente et développe l'analyse relationnelle de concepts pour extraire des règles traduisant tant les caractéristiques propres d'un ensemble d'objets que les liens avec d'autres ensembles. Une première partie développe la théorie mathématique de la méthode, alors que la seconde partie propose trois cas d'application pour étayer l'intérêt d'un tel développement. Les études sont réalisées dans des domaines variés montrant ainsi la polyvalence de la méthode : un premier cas traite l'analyse d'erreur en production industrielle métallurgique, un second cas est réalisé en psycholinguistique pour l'analyse de dictionnaires et un dernier cas montre les possibilités de la méthode en ingénierie de connaissance.

Mots-clés : extraction de connaissance ; fouille de données multi-relationnelles ; analyse relationnelle de concepts ; règles d'association ; interprétabilité.

INTRODUCTION

Aujourd'hui, la production de données est colossale. Bien que les chiffres pour l'année 2020 ne soient pas disponibles au moment de rédiger cette thèse, une étude de BSA, disponible sur le site de l'entreprise Microsoft, affirme que déjà en 2016, 2.5×10^{30} octets de données étaient générés chaque jour, avec une tendance croissante (BSA, 2016). De plus, cette étude révèle que 90% des données de l'humanité alors existantes ont été produites seulement au cours des années 2014 et 2015 (BSA, 2016). Toujours selon BSA, elles sont, entre autres, le nerf de la guerre pour «la croissance économique, pour le développement de l'agriculture, pour les décisions politiques et pour la gestion des énergies renouvelables». Une meilleure gestion des données produites, même sensible, peut avoir un impact considérable. En effet, Microsoft (BSA, 2016) et General Electric (G.E., 2012) mentionnent que les économistes s'accordent pour dire que l'amélioration de la compréhension et la gestion des données de seulement 1% pourrait augmenter le PIB international de 15 billions de dollars américains.

Le milieu industriel produit lui aussi sa part de données grâce à l'utilisation de capteurs, des lignes de commandes des clients ou des commandes aux fournisseurs, ainsi que par l'analyse des notes internes. Actuellement, 90% des grandes entreprises citent les données comme une ressource-clé (BSA, 2016). Leur analyse permet, entre autres, une réduction des coûts en optimisant les achats (G.E., 2012), et les temps de travail des employés (O'Neil, 2016). Dans les sciences dites *expérimentales*, la *méthode scientifique* propose la construction de modèle par l'émission d'une hypothèse, puis sa vérification par l'analyse des données produites par expérimentation (Godfrey-Smith, 2009). Dans le domaine des technologies de

l'information, l'hyper-connectivité, que ce soit par les capteurs de *l'internet des objets* ou par les *média sociaux*, engendre une masse de données considérable qui est analysée par les géants du web. Entre autres, elles servent à créer des systèmes de recommandation et sont utilisées à des fins marketing et publicitaires, comme le montrent les produits vendus par Facebook Analytics et Google Ads. Les données sont donc actuellement utilisées comme indicateurs de la «vérité terrain», et un des défis majeurs est d'appréhender ces données pour passer du stade d'observation à celui d'intuition, puis de l'intuition aux réponses (BSA, 2016).

Pour appréhender de larges quantités de données, un traitement automatique est nécessaire. Un tel traitement doit à la fois permettre la formulation d'hypothèses par la description des tendances d'un jeu de données et la vérification de ces hypothèses qui pourront, le cas échéant, être intégrées en tant que théorie du domaine couvert par les données. On appelle, en sciences des données, *découverte de connaissances* la discipline visant à établir de telles théories (Fayyad *et al.*, 1996). La détection des tendances d'un jeu de données est, quant à elle, appelée *fouille de données* (Fayyad *et al.*, 1996).

La discipline de la fouille de données vise à extraire et utiliser les régularités que l'on peut détecter dans un ensemble de données que l'on peut représenter sous forme d'un tableau, c'est-à-dire un ensemble d'objets décrits par un ensemble d'attributs et de valuations (Ye, 2003). Toutefois, actuellement les ensembles de données volumineux proviennent de sources hétérogènes (par exemple, capteurs et bons de commande) et décrivent des types d'objets différents (par exemple, clients et produits) qui peuvent être en relation (par exemple, le client X achète le produit Y). On qualifie de telles données de *relationnelles* et elles peuvent être représentées par un multi-graphe (Džeroski, 2003). Différentes structures, telles que les bases de données relationnelles ou les *triplestores*, peuvent accueillir de telles données. Parmi les exemples les plus importants de jeux de données

relationnelles, on compte le *web des données liées*, le *Knowledge Graph* de Google et l'ensemble de données du média social Facebook.

Dans cette thèse, on s'intéresse à développer une méthode visant à extraire l'information utilisable (et donc compréhensible) par un expert du domaine depuis lesquelles les données sont extraites. Plus particulièrement, on focalise sur les jeux de données les plus génériques possibles : des données relationnelles qui n'ont pas été annotées. Ainsi, pour exploiter la totalité de l'information disponible, on s'intéresse aux méthodes non supervisées de la *fouille de données multi-relationnelles*. Idéalement, pour qu'un expert puisse attester la validité d'un résultat, il convient d'utiliser une méthode dont on peut retracer et comprendre le comportement (à l'opposé des modèles «boîtes noires»). De plus, bien qu'une méthode ne puisse attester la véracité d'une propriété uniquement par l'exploitation des données (Hernán et Robins, 2020), il est possible de présenter les co-occurences à un expert qui peut les évaluer pour en déduire les liens de causalité régissant les régularités.

L'analyse formelle de concepts (AFC) (Wille, 1982) est une méthode d'analyse mathématique qui a été utilisée, à de multiple reprises, comme cadre pour des tâches de fouilles de données (Valtchev *et al.*, 2004; Huchard, 2019; Kapoor *et al.*, 2020). Pour un jeu de données constitué d'un ensemble d'objets présentant des attributs, elle extrait et organise toutes les abstractions conceptuelles présentes dans le jeu de données au sein d'un treillis complet. L'AFC est une méthode de clustering hiérarchique et multigroupe, l'intégrant aux techniques d'apprentissage non supervisées. L'organisation dans le treillis permet ensuite l'extraction de règles d'association non redondantes (Kryszkiewicz, 2002), présentant des co-occurences pertinentes. Cela fait de l'AFC une méthode de fouille de données non-supervisée particulièrement versatile.

Toutefois, l'AFC est limitée à des applications sur des ensembles d'objets homogènes. L'analyse relationnelle de concepts (ARC) est une extension, à la fouille de données multi-relationnelle, de l'AFC permettant de se libérer de cette contrainte, autorisant l'intégration de jeux de données relationnelles tels que les données liées (Rouane-Hacene *et al.*, 2013). L'ARC organise, pour chaque type d'entité, un treillis de concepts séparé, puis connecte ses treillis au travers d'attributs relationnels, permettant d'intégrer pour chaque objet des propriétés issues des objets qui lui sont connectés, qu'ils soient de même type ou non. Néanmoins, les règles d'association n'ont pas été clairement définies pour l'ARC. La circularité entre les concepts engendre une circularité dans les règles d'association, si la même extraction qu'avec l'AFC est appliquée.

Une des contributions fondamentales de cette thèse est de présenter une méthode pour extraire ces règles, tout en empêchant la production de références à la prémisse dans la conclusion qui entravent leur interprétation. Cette nouvelle forme de règles d'association, si formalisée de manière univoque, permet la découverte d'associations relationnelles non triviales des concepts générés par ARC. En utilisant les notions de fermeture et de génération des concepts, nous montrons comment ces règles d'associations peuvent être générées et en quoi l'ARC produit des règles différentes des méthodes actuelles de fouille pour les données liées, telles que la fouille de graphes (Yan et Han, 2002; Nijssen et Kok, 2004), la fouille de schémas logiques (Józefowska *et al.*, 2008) ou les techniques basées sur des jointures préalables (Narasimha *et al.*, 2011)

Dans le chapitre 1, nous introduisons les ontologies et les règles d'association, des structures de représentation de connaissance. Le chapitre 2 présente, après un rappel de certains prérequis mathématiques, la théorie de l'analyse formelle de concept (AFC). L'AFC est la brique fondamentale de notre méthode, elle permet l'extraction d'une représentation compacte de l'ensemble des règles d'association

d'un jeu de données. Ces derniers sont néanmoins limités à des descriptions d'objets en fonction de caractéristiques propres. Le chapitre 3 décrit l'analyse relationnelle de concept (ARC), une extension de l'AFC permettant de prendre en considération les liens entre les objets. Toutefois, l'ARC perd la simplicité d'interprétation des règles d'association produite par l'AFC. On conclut la première partie de cette thèse par le chapitre 4, qui étudie les limites de l'ARC et propose un consolidation de la théorie pour allier la lisibilité des règles de l'AFC et la compréhension des liens inter-objets prévue par l'ARC. La partie II est composée de trois chapitres indépendants reflétant chacun un cas d'utilisation de l'ARC dans la forme proposée au chapitre 4. Le chapitre 5 propose un cas d'étude en production industrielle. On y montre que l'extraction de règles d'association révèle les liens entre les caractéristiques des pièces manufacturées, par moulage d'aluminium, et les problèmes de production. Le chapitre 6 étudie un cas en psycholinguistique. On y analyse les corrélations entre la structure de différents dictionnaires et les mesures psycholinguistiques des mots qui les composent. Le chapitre 7 présente l'ARC comme méthode pour l'ingénierie de connaissance pour diverses tâches : la restructuration d'un schéma de données, la détection d'erreur de typage et l'évaluation du schéma par les données. Finalement, on conclut cette thèse en suggérant plusieurs pistes de recherche pour continuer à développer l'ARC.

Première partie

Théorie

CHAPITRE I

EXTRACTION DE CONNAISSANCE

Historiquement, formaliser la connaissance dans des systèmes informatiques avait pour but de concevoir une structure fondamentale, sur laquelle pourrait être construite une intelligence artificielle universelle de résolution de problèmes, le *general problem solver* (Newell *et al.*, 1959). Le succès dans l'atteinte de cet objectif, trop ambitieux, a été mitigé. En effet, ce système, ratissant trop large, ne pouvait raffiner ses stratégies algorithmiques, et se retrouvait victime de l'explosion combinatoire (Russell et Norvig, 2002). En quelques années, la communauté scientifique de l'intelligence artificielle s'est tournée vers la conception de systèmes experts (Russell et Norvig, 2002). Ces derniers visent à atteindre au moins les capacités humaines pour la résolution d'une tâche spécifique ; mais ils n'ont pas vocation à être employés pour tout problème. Parmi les domaines célèbres où ces systèmes experts ont été déployés, on compte, entre autres, les problèmes de planification, de diagnostic et de classification (Baader *et al.*, 2003; Russell et Norvig, 2002).

Plus récemment, le développement significatif du *web* sémantique a encouragé le développement de jeux de données liées ouvertes : en mars 2019, DBPédia l'un des jeux de données les plus volumineux contenait 9.5 milliards de liens entre ses données ; et on pouvait dénombrer 1 239 jeux de données ouverts et inter-

connectés entre eux (lod-cloud.net, 2019). Ces ensembles de données volumineux permettent le développement d'une nouvelle génération de systèmes experts (Russell et Norvig, 2002). Toutefois, pour exploiter ces ensembles conjointement, un besoin d'interopérabilité doit préalablement être comblé. Les ontologies informatiques, schémas de bases de connaissance, ont donc émergé pour y répondre (Arp *et al.*, 2015).

Les systèmes experts sont composés de deux briques fondamentales : un raisonneur et une base de connaissances (Russell et Norvig, 2002). Pour résoudre un problème spécifique, il faut manipuler la connaissance disponible adéquatement. Pour cela, un raisonneur combine les éléments spécifiques d'une situation par différents mécanismes, notamment l'inférence (Baader *et al.*, 2003). Pour manipuler la connaissance, il faut, bien évidement, définir un formalisme permettant une manipulation par la machine. Celui-ci doit être assez expressif pour englober toute forme de connaissance, et assez précis pour être compatible avec le traitement rigide et binaire d'une machine. Plusieurs logiques répondent à de tels critères : logique du premier ordre, statistique, logique floue, etc. (Russell et Norvig, 2002).

L'ingénierie de la connaissance est la discipline qui consiste à concevoir la base de connaissances. Elle est composée de sept étapes (Russell et Norvig, 2002) :

1. Identifier la tâche et définir les limites de la connaissance à représenter.
2. Acquérir les connaissances.
3. Formaliser la connaissance rassemblée.
4. Encoder la connaissance formalisée.
5. Encoder la description des problèmes à résoudre.
6. Appliquer le raisonneur.
7. Corriger la base de connaissances (retour à l'étape 1).

La présente thèse se focalise sur la tâche de l'extraction de connaissances (étape

2) pour des bases de connaissances compatibles avec le format OWL2, le langage d'implémentation des ontologies recommandé par le W3C (W3C, 2012). Ainsi, dans ce chapitre, nous nous attelons, dans un premier temps, à présenter différentes définitions de la connaissance, et justifions notre choix de l'une d'entre elles en section 1.1. Dans la section 1.2, nous présentons les ontologies avec ses formalisations dédiées. La section 1.3 présente différentes tâches d'extraction de connaissance. Nous mettons l'accent sur la tâche de l'extraction de règles d'association relationnelles, puisque celle-ci se trouve au cœur du processus décrit dans les prochains chapitres de la thèse. Finalement, la section 1.4 conclut ce chapitre en mettant l'analyse relationnelle de concepts en relief par rapport à l'ingénierie de connaissance.

1.1 Définition de la connaissance

En sciences de l'information, les termes de *donnée*, d'*information* et de *connaissance* sont souvent évoqués. Dans ce domaine, ces termes suivent les définitions du modèle *Data-Information-Knowledge-Wisdom* (DIKW), souvent représenté par une pyramide (cf fig. 1.1).

Ce premier modèle est aujourd'hui prédominant en sciences de l'information. Toutefois, dans un article de référence [1] du domaine de management de la connaissance, Alavi et Leidner soutiennent que d'autres vues sont tout autant admissibles (Alavi et Leidner, 2001), notamment celui de Tuomi. Ce dernier défend une vision permutée du premier modèle en affirmant que les données sont postérieures à la connaissance : chaque donnée est extraite par un processus lui-même influencé par une connaissance préalable et l'information catégorise un ensemble de données au travers d'une connaissance préalable (Tuomi, 1999).

1. avec plus de 13 000 citations

Figure 1.1: Pyramide DIKW (longlivetheux, 2005)

Dans cette thèse, nous nous focalisons toutefois sur la définition de connaissance suivant le modèle *Data-Information-Knowledge* (la partie *Wisdom* est hors du champ d'étude de cette thèse). Puis, nous nous plaçons dans un des cadres possibles offerts par ce modèle.

Avant de présenter le modèle de manière plus stricte, illustrons, au travers d'un exemple, les différentes catégories : *données*, *information* et *connaissance*.

Exemple 1. *Imaginons le cas d'un système d'arrosage automatique et intelligent. Le système a pour but de distribuer de l'eau à une plante uniquement lorsque nécessaire.*

Supposons maintenant les faits suivants : Mardi 16/02, la température est de 21^oC*, la terre est sèche ; Mercredi 17/02,* 15^oC*, humide ; Jeudi 18/02,* 20^oC*, sèche ; Vendredi 19/02,* 17^oC*, humide ; Mardi 23/02 ;* 21^oC*, sèche. Ces faits, individuellement, ne sont que de simples relevés, sans intérêt particulier. On les nomme* données.

En agrégeant ces données, on peut retrouver de l'information telle que : «les mardis la terre est sèche», «lorsque la température est au dessus de 19^oC*, la terre est sèche», etc.*

Enfin, en filtrant l'information, on peut en extraire la connaissance que doit inté-

grer le système : «s'il fait plus de 19^oC*, arroser la plante dans la journée».*

Cet exemple est une illustration possible du modèle, mais d'autres sont tout autant admissibles. Zins a collecté les définitions de *donnée*, *information* et *connaissance* produites par 57 chercheurs de 16 nationalités différentes (Zins, 2007). En compilant ces résultats, l'auteur a pu montrer que la définition du modèle *Data-Information-Knowledge* était loin d'être unique et universelle. Il a notamment pu extraire de ces réponses cinq modèles distincts. Tous ces modèles s'accordent pour dire qu'un fait, extérieur à tout individu (comme la température), est une donnée. Certains, mettent aussi en évidence ce qu'on pourrait qualifier de la *perception d'un fait* (par exemple, «il fait froid»), comme donnée.

Pour ce qui est de la notion d'information, les points de vue divergent déjà considérablement. Certains considèrent que l'information est une contextualisation des données : *Information is what context creates/gives to data* (Zins, 2007) ; alors que d'autres considèrent l'information comme une intériorisation des données : *Information is an organism's [. . .] inferential frame that guides the selection of data for its own further development or construction* (Zins, 2007). Enfin, pour la notion de connaissance, encore une fois, les points de vue s'opposent. Bien que tous considèrent que la connaissance est interne à un agent («*Knowledge is what was understood and evaluated by the knower*» (Zins, 2007)), certains considèrent que l'extériorisation est possible («*Knowledge can be given physical representation (presence) in the material products (technology) thereof (books, film, speech, etc.)*» (Zins, 2007)). Zeleny soutient que l'extériorisation est impossible et va jusqu'à affirmer que la connaissance ne peut se démontrer qu'au travers d'actions («le seul moyen de signaler qu'on sait faire un gâteau est d'en produire un») (Zeleny, 2005). Toute verbalisation de la connaissance («je sais faire un gâteau») ne serait que la production d'information. On peut aussi y lire «*The only bad thing about knowledge is that many people, experts and laymen alike, treat knowledge as some*

sort of higher-level information : extended, synthetic, advanced, complex, etc. , but still information. » : la confusion entre connaissance et information semble être, selon Zeleny, une méprise assez générale (Zeleny, 2005).

Dans cette thèse, on utilise les termes *données*, *information* et *connaissance* tels que définis dans l'article (Fayyad *et al.*, 1996), prédominant [2] dans le domaine des sciences de l'information. Il nous sert de référence dans le vocabulaire employé. Dans cet article, on peut lire «*Historically, the notion of finding useful patterns in data has been given a variety of names, including data mining, knowledge extraction, information discovery, information harvesting, data archaeology, and data pattern processing*» et «*Given these notions, we can consider a pattern to be knowledge if it exceeds some interestingness threshold*». Ainsi nous partons du postulat que l'extraction de connaissances consiste à découvrir un ensemble d'informations, sous forme de régularités. Cet ensemble doit être organisé de sorte à présenter uniquement les plus pertinentes, pour une intégration optimale au système de connaissance.

Pour revenir à notre exemple sur l'arroseur automatique, l'extraction de connaissance est un processus qui produit un résultat intermédiaire, entre l'ensemble des informations («les mardis la terre est sèche», «lorsque la température est au dessus de 19^oC, la terre est sèche», . . .), et l'ensemble de la connaissance («s'il fait plus de 19^oC, arroser la plante dans la journée»). L'ensemble produit doit contenir des informations, mais doit être restreint à celles utiles à l'acquisition des connaissances (dans ce cas, on aurait uniquement «lorsque la température est au-dessus de 19^oC, la terre est sèche»).

---

2. Plus de 10000 citations.

1.2 Représentation de la connaissance

Dans la littérature, on trouve de multiples formalismes logiques et de représentation de la connaissance. Le chapitre 4 de (Baader *et al.*, 2003) présente différents formalismes logiques tels que les logiques modales (Schild, 1991), les restrictions de la logique du premier ordre aux fragments décidables (Andréka *et al.*, 1998), les modèles de base de données, ainsi que des représentations de la connaissance utilisées dans le domaine de l'intelligence artificielle, tels que les réseaux sémantiques (Brachman, 1979), le système de *frames* (Hayes, 1981) et les graphes conceptuels (Sowa, 1983). Toutefois, avec l'émergence du *web* des données, le W3C, l'organisme de standardisation international des technologies du *web*, recommande, à des fins de réutilisation et d'interopérabilité, le développement d'ontologies au format OWL2 (W3C, 2012). OWL2 est un langage basé sur la logique de description $\mathcal{SROIQ}$. Ainsi, dans cette section nous présentons, dans un premier temps, les langages de formalisation des ontologies, notamment la logique de description $\mathcal{SROIQ}$ (Horrocks *et al.*, 2006), avant de présenter la définition d'ontologie d'une base de connaissances.

1.2.1 Formalisation

La logique du premier ordre est composée de *fonctions*, de *variables*, de *constantes*, ainsi que de *prédicats* qui servent à exprimer les propriétés et relations de ces variables et éléments. Par convention, on note les variables et les fonctions en majuscules, alors que les constantes et prédicats commencent par une minuscule.

Un *terme* est une variable, une constante, ou la composition d'un de ces éléments dans une fonction. Les prédicats, adjoints de leurs termes, forment des *atomes*, la brique fondamentale de ce formalisme. L'abstraction au-dessus de l'*atome* qui

englobe sa valuation binaire (vrai ou faux) est appelé un *littéral*. Les prédicats, tout comme les fonctions, peuvent lier plusieurs termes. Le nombre de liens est appelé l'*arité* du prédicat. L'exemple 2 illustre ces définitions. En logique du premier ordre, ces atomes sont combinés en formules.

Exemple 2. *Prenons (1) les variables W, X, Y et Z, (2) la fonction frère, qui, à une variable représentant un individu, associe le frère de l'individu, (3) une constante adam, et (4) le prédicat amis. Alors amis(W, X, Y, Frère(Z), adam) est un atome dont W, X, Y, Frère(Z) et adam sont les termes. Le prédicat amis est dit d'arité 5 et attribue la propriété d'être* amis *à ces cinq termes. Lorsqu'on affecte une valeur vrai ou faux à l'atome, on parle de littéral.*

Les symboles de négation ($\neg$), de conjonction ($\wedge$), de disjonction ($\vee$), d'implication ($\rightarrow$) et d'équivalence ($\equiv$), sont les connecteurs des formules. Il existe ($\exists$) et quel que soit ($\forall$) sont appelés quantificateurs.

Formellement, un atome est une formule. De plus, si e_1, e_2 sont des formules alors $\neg e_1, e_1 \wedge e_2, e_1 \vee e_2, e_1 \rightarrow e_2, e_1 \equiv e_2, \forall x e_1, \exists x e_1$ sont des formules (x étant une variable).

La logique du premier ordre est particulièrement expressive. Toutefois, elle présente un défaut majeur pour ce qui est de la connaissance que l'on peut formaliser depuis les données : elle n'est pas une logique décidable. C'est-à-dire qu'il est possible de concevoir des formules, pour lesquelles il est impossible de décider si elles sont vraies pour toute donnée. L'indécidabilité de cette logique est un problème majeur lorsqu'implémentée dans un système informatique. On pourrait, entre autres, demander de vérifier une formule qui déclencherait un traitement infini, sans avoir moyen de savoir si le traitement est effectivement infini, ou extrêment long.

Toutefois, certains fragments (ou restrictions) de la logique du premier ordre sont décidables. Parmi les fragments décidables, on trouve notamment *la restriction à deux variables*, c'est-à-dire où tous les prédicats et fonctions ne présentent que des arités 1 ou 2. Les logiques de descriptions sont des logiques basées sur la logique du premier ordre acceptant certains de ses connecteurs et quantificateurs. Elles sont (généralement) restreintes à deux variables et décidables (Baader *et al.*, 2003).

Il existe de multiples logiques de description. Dans cette sous-section, comme nous ne pouvons pas toutes les décrire exhaustivement, nous nous concentrons sur la logique $\mathcal{SROIQ}$ (Horrocks *et al.*, 2006). Ce choix est motivé par le fait que $\mathcal{SROIQ}$ a été retenue par le W3C pour les spécifications d'OWL2, le langage de définition recommandé pour les ontologies.

Les logiques de description se basent avant tout sur des concepts et rôles atomiques. Ceux-ci peuvent être respectivement dénotés en logique du premier ordre par les prédicats d'arité 1 (concepts) et 2 (rôles). $\top$ et $\bot$, sont les concepts universellement vrai et faux. Ainsi, $\top$ est le concept qualifiant toutes les instances possibles alors que $\bot$ ne qualifie aucune instance.

Avec ces briques de base, on peut maintenant décrire la logique $\mathcal{SROIQ}$. Si C et D sont des concepts, alors l'intersection $C \sqcap D$, l'union $C \sqcup D$, et le complémentaire $\neg C$ sont des concepts. De plus, pour un rôle R, les restrictions existentielle $\exists R.C$, universelle $\forall R.C$, au moins $\geq n\ R.C$ et au plus $\leq n\ R.C$, ainsi que la réflexivité locale décrivant l'ensemble des instances en relation avec elles-mêmes, notée $\exists R.Self$, sont également des caractérisations de concepts.

Une des particularités de $\mathcal{SROIQ}$ est que, si une relation R est définie, alors sa relation R^-, dite *inverse*, l'est aussi. En logique du premier ordre, cela revient à dire que, si on a le predicat $R(x, y)$, alors on a également $R^-(y, x)$.

L'intérêt d'un tel formalisme est d'une part son expressivité, d'autre part le fait qu'il soit décidable (Horrocks *et al.*, 2006).

1.2.2 Représentation de la connaissance

Une base de connaissances est un ensemble de *phrases* exprimées dans un langage de représentation de la connaissance. Les *phrases* qui sont considérées immédiatement comme vraies, sans être inférées, sont appelées *axiomes* (Russell et Norvig, 2002). Ces *phrases* peuvent soit décrire le schéma de la base de connaissances, c'est-à-dire présenter un ensemble de concepts et les relations entre eux, soit décrire les instances, que ce soit les liens entre elles ou leurs liens aux éléments du schéma. Le schéma d'une telle base est appelé *ontologie* de la base de connaissances (Russell et Norvig, 2002).

Les ontologies étant définies à partir d'une taxonomie, on reprend la définition de cette dernière avant d'introduire les ontologies.

Définition 1. *Une* taxonomie *est une hiérarchie composée de termes décrivant des types (universels ou classes) reliés entre eux par une relation de généralisation (Arp* et al.*, 2015).*

Définition 2. *Une* ontologie *est un outil de représentation. Elle comprend une taxonomie fondamentale, un ensemble d'universaux et de classes combinant les éléments de cette taxonomie, ainsi que des relations entre ces éléments (Arp* et al.*, 2015).*

Pour permettre une bonne interopérabilité et réusabilité des ontologies, on peut définir quatre niveaux d'ontologies : de haut niveau, intermédiaire, de domaine et d'application (Arp *et al.*, 2015).

Les ontologies d'application sont créées dans le but de répondre à une tâche très

spécifique. Elles ont un domaine de définition très concret et sont peu réutilisables. Les ontologies de domaine représentent, comme leur nom l'indique, un domaine en particulier. Elles décrivent de manière exhaustive les entités et leurs relations. Par exemple, on peut mentionner l'ontologie de domaine CVDO (*CardioVascular Disease Ontology*). Les ontologies de niveau intermédiaire sont plus larges et sont utilisables par plusieurs ontologies de domaine. Par exemple, on peut compter DO (*Disease Ontology*) sur laquelle se repose CVDO. Les ontologies de haut niveau sont indépendantes du domaine et ont pour but de caractériser la structure d'une ontologie. Par exemple, BFO (*Basic Formal Ontology*) vise à définir le comportement d'ontologies décrivant un domaine réel (comme la DO) (Arp *et al.*, 2015).

En pratique, pour être intégrées à un système informatique, les ontologies sont encodées en logique du premier ordre. Par exemple, l'ontologie de haut niveau DOLCE, visant caractériser les ontologies s'appuyant sur le langage naturel et le sens commun, est écrite complètement en logique du premier ordre (Masolo *et al.*, 2003). Mais pour des raisons de performance et notamment de décidabilité, il est recommandé de les écrire en OWL2 (W3C, 2012; Arp *et al.*, 2015).

1.3 Extraction de connaissance

«La découverte de connaissance est le processus non trivial d'identifier des régularités dans les données qui sont valides, nouvelles, potentiellement utiles et ultimement compréhensibles» (Piatetsky-Shapiro *et al.*, 1996). Il existe principalement trois objectifs de haut niveau pour lesquels il convient de mettre en place un tel processus : *vérifier une hypothèse*, *créer un modèle de prédiction* ou *créer un modèle de description* (Fayyad *et al.*, 1996). Un *modèle de prédiction* vise à estimer certaines valeurs pour de nouvelles données, ou l'évolution de certaines va-

leurs pour des données existantes (Fayyad *et al.*, 1996). Le *modèle de description* vise à détecter et décrire les régularités (groupes et tendances) présentes dans les données (Fayyad *et al.*, 1996). Pour réaliser ces objectifs, plusieurs phases sont nécessaires : sélection des données, prétraitement, transformation, fouille de données, interprétation et évaluation de l'information (Fayyad *et al.*, 1996).

Nous nous intéressons ici à l'extraction de connaissances pouvant construire des modèles de description, notamment les ontologies. Ainsi, dans une première sous-section nous présentons la fouille de données (multi-relationnelle), puis nous parlons d'extraction d'ontologie. Finalement, nous décrivons les règles d'association, une des formes d'information particulièrement directe à interpréter, conjointement avec des méthodes d'extraction dédiées.

1.3.1 Fouille de données multi-relationnelles

L'extraction des connaissances depuis un jeu de données demande d'analyser les données disponibles pour en extraire les régularités, constituant l'information. La discipline d'analyse automatisée des jeux de données informatisés s'appelle *fouille de données* (Fayyad *et al.*, 1996). La discipline de fouille de données regroupe un ensemble de méthodes étudiant les régularités présentes dans un jeu de données homogène (Ye, 2003). Or aujourd'hui, comme nous l'avons soulevé dans l'introduction, les jeux de données sont de plus en plus hétérogènes et structurés selon un paradigme relationnel.

Pour exploiter des jeux de données relationnelles, trois possibilités sont envisageables :

— La fouille de données classique, qui considère l'extraction de structures pour un seul type d'entité. On peut l'utiliser sur chaque type d'entité séparement et ne pas considérer les liens. Toutefois, l'intégration de ces liens peut être

émulée par la création d'un type d'entité générique qui correspondrait à un produit cartésien de tous les types existants.

— La fouille de graphes (Yan et Han, 2002; Nijssen et Kok, 2004) ne s'intéresse, quant à elle, qu'à l'aspect structurel du graphe de relation. Cette discipline vise à générer pour un ensemble de graphes, un ensemble de sous-graphes fréquemment trouvés dans l'ensemble de graphes de départ.

— La fouille de données multi-relationnelles (Džeroski, 2003) est un sous-domaine plus structuré de la fouille de données qui vise à observer les données en considérant plusieurs tables, chacune représentant un type d'entité différent, ainsi que les liens entre les données de différentes tables.

On s'intéresse ici, à cette dernière alternative. De multiples méthodes de la FDMR ont été proposées, répondant aux versions étendues des problèmes de la fouille de données *classique* (Džeroski, 2003). Parmi ces applications on trouve, notamment, la régression (Cai *et al.*, 2005), la classification (Frank *et al.*, 2007), la découverte d'associations (Galárraga *et al.*, 2013) et le *clustering* (Nergiz *et al.*, 2008).

Bien évidemment, il n'y a pas de façon unique d'étendre un algorithme de fouille de données au paradigme relationnel. Par exemple, Mistry et Thakkar proposent de créer un classificateur par table, puis de combiner les différents résultats obtenus au travers d'un réseau bayésien (Mistry et Thakkar, 2014). Dans la même veine, Bina et al. utilisent des arbres de décisions, un par type d'objets, dont ils combinent les résultats à postériori, selon les relations entre les classes qui existent (Bina *et al.*, 2013). Dans une optique inverse, Galárraga et al. ainsi que Bühmann et al. intègrent l'information relationnelle en amont du traitement (Galárraga *et al.*, 2015) et (Bühmann *et al.*, 2016).

1.3.2 Extraction d'ontologie

L'extraction d'ontologie vise à concevoir automatiquement, à partir de données, une ontologie. Sabou et al. soulèvent toutefois le problème majeur suivant : il n'existe pas encore de solution à l'évaluation d'une méthode d'extraction d'ontologie (Sabou *et al.*, 2005). Actuellement, les méthodes sont évaluées en comparant leurs résultats avec une ontologie établie manuellement, en anglais on les qualifie de *gold standard*. Mais une ontologie de domaine réalisée par une méthode automatique peut être évaluée (manuellement) plus pertinente et précise que le *gold standard* par les concepteurs de ce dernier (Sabou *et al.*, 2005). Il existe principalement deux façons de procéder pour concevoir une ontologie : manuelle, par des experts-domaine, ou automatique, depuis une source de données (Hazman *et al.*, 2011). Toutefois, comme les méthodes ne peuvent pas être évaluées de manière automatique, un processus possible est de concevoir une proto-ontologie automatiquement, de la transmettre à un expert du domaine pour trancher sur les choix incertains. On qualifie un tel traitement de *semi-automatique*. Toutefois, Sabou et al. soulèvent notamment le cas suivant : faut-il signaler les synonymes par une relation de synonymie ou regrouper tous les synonymes dans un même concept (Sabou *et al.*, 2005) ?

Que la méthode soit manuelle, automatique ou semi-automatique, la procédure doit prendre en compte le format des données : non-structurées (par exemple, texte en langage naturel), semi-structurées (par exemple, un texte en langage naturel dont les mots et sections sont marqués par les étiquettes), ou structurées (par exemple, base de données). Le rapport d'Hazman et al. survole différentes techniques pour l'extraction depuis les données non-structurées et semi-structurées, des approches statistiques, des techniques de traitement automatique du langage, et diverses techniques de fouille de données (Hazman *et al.*, 2011). D'autres

chercheurs se sont intéressés aux données structurées. De nombreux outils sont disponibles et référencés (Unbehauen *et al.*, 2012). Par exemple, l'outil RDB-to-Onto (Cerbah, 2008) vise à extraire une ontologie depuis une base de données relationnelle en s'appuyant tant sur le schéma que sur les données pour découvrir et enrichir des concepts.

Plusieurs de ces outils sont notamment basées sur un processus de règles d'association. Ils ont été utilisées tant sur des données non-structurées pour étudier les corrélations entre les mots d'un texte et leurs catégories dans une taxonomie que sur des données structurées (Maedche et Staab, 2000), tant pour la fusion d'ontologies que pour la découverte de relation de subsomption entre les classes (David, 2007).

1.3.3 Extraction des règles d'association

Bien qu'on ne puisse pas extraire un modèle causal depuis des données uniquement (Hernán et Robins, 2020), on peut s'en rapprocher en mettant en avant les corrélations entre les différentes variables. Ces corrélations, combinées à un modèle ou au sens commun, peuvent servir de base à la conception d'un nouveau modèle ou la modification d'un modèle préalablement existant.

Les règles d'associations sont un formalisme qui permet d'exprimer des régularités de la forme «$X\%$ des objets qui possèdent les propriétés de A ont aussi les propriétés de B» (Agrawal *et al.*, 1996). Par exemple, on peut avoir une règle de la forme «87.3% des clients qui achètent des couches un jeudi achètent aussi de la bière». Formellement, une règle d'association se présente sous la forme $A \rightarrow B \setminus A$, où A et B sont des ensembles de propriétés (dans l'article original des *items* (Agrawal *et al.*, 1993)). En logique de description, elle se traduit par l'énoncé suivant : pour un individu x, si l'assertion $A(x)$ se trouve dans la base de connaissances, alors

$B(x)$ également. Ces règles, toutefois, admettent une certaine tolérance à l'erreur. On se donne alors deux mesures de qualité d'une règle : le *support* et la *confiance*.

Le *support* matérialise l'importance d'une règle en terme de représentativité dans une base. Il est noté $supp(A \rightarrow B \setminus A)$ ou $\sigma(A \rightarrow B \setminus A)$ et correspond au nombre d'individus d'une base de connaissances qui sont à la fois des représentants du concept A et du concept B. Formellement :

$$supp(A \rightarrow B \setminus A) = |\{x|A(x) \wedge B(x)\}| \tag{1.1}$$

Alternativement, on peut aussi définir le support, non pas en terme absolu, mais relativement au nombre d'individus existants. La *confiance*, quant à elle, mesure la validité d'une règle, ou encore la corrélation entre A et B. Elle dénote le ratio entre le nombre d'individus x vérifiant $B(x)$ par rapport à ceux qui vérifient $A(x)$. Ainsi, la confiance se donne par l'expression suivante :

$$conf(A \rightarrow B \setminus A) = \frac{|\{x|A(x) \wedge B(x)\}|}{|\{x|A(x)\}|} \tag{1.2}$$

Lorsque la confiance d'une règle vaut 1, on la qualifie de règle exacte ou règle d'implication ; si la confiance est inférieure, on parle de règle approximative. Ces mesures, de confiance et de support, peuvent être combinées pour définir des mesures d'intérêt qui servent à séparer l'information superflue de la connaissance (Fayyad *et al.*, 1996). Toutefois, il n'existe pas de consensus général sur la notion de mesure d'intérêt (Geng et Hamilton, 2006).

Voyons maintenant différents algorithmes de la littérature, focalisés sur l'extraction de règles d'association, dans un contexte de base de connaissances, c'est-à-dire un contexte relationnel, avec différents rôles et concepts possibles.

Les règles d'association relationnelles ont été sujettes à de nombreuses études et

méthodes. Parmi les méthodes d'extraction les plus connues, on trouve AMIE, définie dans (Galárraga *et al.*, 2013). Cette approche a été développée pour faire de la prédiction de liens dans un graphe de connaissances. Par conséquent, cette méthode ne s'intéresse qu'aux règles ayant une conclusion représentant un lien unique. En logique des prédicats, cela se traduit par une clause de Horn dont la conclusion ne contient qu'un atome binaire (d'arité 2). Cette limitation n'est pas problématique pour la tâche de prédiction de liens ; mais, toutefois, elle restreint les connaissances possibles à l'extraction. Plusieurs autres méthodes suivent ce schéma.

À haut niveau, ces algorithmes se traduisent par la procédure dénotée par l'algorithme 1. Cette dernière peut être assimilée à une heuristique gloutonne, que nous décrivons ici. L'algorithme commence par se munir d'un ensemble initial de règles candidates (ligne 1). Tant que cet ensemble n'est pas vide (ligne 2), une règle en est sélectionnée et extraite (ligne 3). Ensuite, la règle passe dans une phase de traitement. Si elle est intéressante (étant donné ce qui a déjà été découvert), on l'ajoute à l'ensemble des règles produites (ligne 5 à 7). Puis, intéressante ou non, on étend cette règle en utilisant les prédicats disponibles dans la base de connaissances. Pour l'étendre, on commence par générer les voisins immédiats de la règle (ligne 8). Puis, un filtre est appliqué, pour ne conserver que les voisins prometteurs (ligne 10). Les règles voisines restantes sont alors ajoutées à l'ensemble des règles candidates (ligne 11). Lorsqu'il n'y a plus de règles candidates, l'algorithme présente les règles validées (ligne 15).

Bien évidemment plusieurs variations de cette méthode existent. Elles concernent donc quatre éléments : (1) l'initialisation (ligne 1), (2) le test de sélection (ligne 5), (3) la génération des voisins (ligne 8) et (4) leur filtrage (ligne 10).

L'algorithme Warmr, défini dans (Dehaspe et Toivonen, 1999) se base sur cette

Algorithme 1 : Algorithme de fouille par extension de règles

Résultat : Ensemble de règles d'association

```
1  début
2      Candidats ← REGLESINITIALISATION()
3      RèglesValides ← ∅
4      tant que (!Candidats.VIDE()) faire
5          r ← Candidats.SELECTION()
6          si TESTINTERET(r) alors
7              RèglesValides.AJOUTE(r)
8          fin
9          V ← GENERERVOISINS(r)
10         pour v ∈ V faire
11             si PERTINENT(v) alors
12                 Candidats.AJOUTE(v)
13             fin
14         fin
15     fin
16     retourner RèglesValides
17 fin
```

logique générique et cherche une règle dont la conclusion est composée d'un atome spécifié par l'utilisateur. (1) L'ensemble des candidats est initialisé en ne contenant que la règle vide. (2) Pour chaque règle candidate, si son support dépasse un seuil fixé par l'utilisateur, elle est considérée valide. (3) WARMR utilise une grammaire spécifiée en paramètre pour générer des règles voisines (des règles de prémisses plus longues). (4) Les voisins qui sont équivalents à un candidat ainsi que ceux plus spécifiques qu'une règle considérée infréquente sont écartés. Au final, l'algorithme renvoie les règles fréquentes.

Dans AMIE (Galárraga *et al.*, 2013), une règle ne peut contenir, en conclusion,

qu'un atome binaire. Une règle de la forme $A_2, ..., A_n \rightarrow A_1$ est représentée par un n-uplet d'atomes $(A_1, ..., A_n)$. De plus, AMIE utilise une variation de la confiance adaptée à l'hypothèse de monde ouvert, appelée *PCA-confidence*. Elle intègre à l'équation de la confiance une estimation des instances qui ne sont pas couvertes par la base. Décrivons maintenant les variations de la méthode. (1) L'ensemble des candidats est initialisé en ne contenant que la règle vide. (2) Le test de validation considère trois éléments pour décider de présenter une règle dans l'ensemble de sortie. Premièrement, elle doit vérifier la condition d'être close, c'est-à-dire toutes les variables de la règle apparaissent au moins deux fois. Ensuite, elle doit vérifier que la prémisse dépasse un certain *support*. Enfin, la règle au complet doit passer un certain seuil de *PCA-confidence*. Seulement la vérification de ces trois contraintes permet de la déclarer d'intérêt. (3) Pour générer des voisins, AMIE rajoute un atome binaire à la prémisse de la règle source. Cet atome doit contenir une variable qui existait déjà dans la formule. Si on adopte une représentation sous forme de graphe de la base de connaissances, cela revient à étendre de manière connexe le sous-graphe représenté par la règle. Le second élément de l'atome ajouté devra être une variable, instanciée ou non, nouvelle ou non. (4) Enfin, si un voisin généré représente une règle équivalente (par permutation des atomes de la prémisse) à une règle déjà candidate, il est ignoré. Les autres sont ajoutés à la liste des règles candidates.

AMIE+ (Galárraga *et al.*, 2015) est une optimisation de AMIE. Certaines variations sont à noter. (1) La file est instanciée avec tous les atomes possibles de la base plutôt qu'avec la règle vide. (2) Le calcul du support est optimisé. (3) La taille des règles voisines est bornée ; et une règle de *PCA-confidence* de 100% ne produit plus de voisin.

L'outil DL-learner (Lehmann et Hitzler, 2007a) est basé sur un algorithme (synthétisé dans l'algorithme 2) qui, contrairement aux méthodes précédentes, est basé

sur un apprentissage supervisé. L'objectif de cet algorithme est d'adapter les techniques de programmation logique inductive (ILP) aux logiques de description, en particulier lorsque les faits sur les instances sont facilement accessibles mais que le schéma les structurant est manquant, ou trop peu expressif pour permettre un raisonnement automatisé. DL-Learner cherche donc à trouver des explications logiques pour certaines données (principe de l'ILP). Il se place dans le cadre où une base de connaissances plus ou moins raffinée est présente, et qu'elle s'accompagne d'observations positives (directement présentes ou inférables) et d'autres, négatives. La méthode vise donc à étendre la base de connaissances avec une formule expliquant les observations. Pour cela, DL-Learner cherche une hypothèse H qui explique les observations. C'est-à-dire qu'idéalement, l'hypothèse, combinée à la base de connaissances, permet d'inférer toutes les observations positives et aucune observation négative. Cette hypothèse décrit une probable classe manquante de la base. Pour ce faire, il utilise un opérateur de raffinement (généralisation ou spécialisation de concept, tel que la subsomption et la supsomption). L'opérateur est choisi pour être compatible avec un travail sur les logiques de description $\mathcal{ALC}$ (Lehmann et Hitzler, 2007b) et $\mathcal{ALCQ}$ (Lehmann et Hitzler, 2010).

Détaillons l'algorithme 2. Un noeud racine est initialisé avec une formule caractérisant tous les faits, positifs comme négatifs. Récursivement, des sous-noeuds vont être ajoutés dans une structure arborescente, jusqu'à trouver une formule qui maximise le nombre de faits positifs et minimise le nombre de faits négatifs couverts. Une itération consiste à sélectionner la feuille avec le meilleur score, créer toutes les formules qui étendent la feuille sélectionnée d'un concept atomique par conjonction. Pour chaque nouvelle formule, on calcule sa qualité. Si elle dépasse un certain seuil, et que la formule n'est pas une permutation d'un autre noeud de l'arbre, on crée une feuille qu'on rattache à la formule d'origine. Dans sa première version, aucun faux négatif n'était accepté, c'est-à-dire que seules les formules

Algorithme 2 : Algorithme de DL-Learner

Données : base de connaissances, Ensemble d'observations positives et négatives

Résultat : Ensemble de concepts caractérisant les observations

```
1  début
2      ST ← InitArbreRecherche()
3      tant que ST ne contient pas de noeud performant faire
4          N ← ST.meilleurNoeud()
5          N.etendre() :
6          début
7              pour A : atome de la base de connaissances faire
8                  M ← Noeud()
9                  M.formule ← N.formule ⊓ A
10                 si M.pertinent() alors
11                     N.ajouterFils(M)
12                 fin
13             fin
14         fin
15     fin
16     retourner ST.feuilles()
17 fin
```

couvrant *tous* les exemple positifs étaient acceptées et généraient un nouveau noeud (Lehmann et Hitzler, 2007b). Ainsi, l'algorithme raffinait les formules uniquement pour minimiser le nombre d'observations négatives couvertes. Le score correspond, dans cette version, au nombre de faits négatifs écartés par la formule.

La seconde version relaxe un peu cette contrainte : elle introduit un seuil de bruit autorisé, fixé au préalable par l'utilisateur (Lehmann et Hitzler, 2010). Dans cette version, la notion du score est étendue. Elle prend en compte : (1) le nombre

d'instances mal classées, positives comme négatives, (2) l'amélioration engendrée par l'extension de la formule «parent», ainsi que (3) la taille de la formule du noeud, en pénalisant les formules plus longues. Les impacts de ces trois mesures sont pondérés par des paramètres fixés par l'utilisateur.

Les algorithmes de (Quinlan, 1990) et (Zeng *et al.*, 2014) s'inscrivent dans la même logique, mais génèrent un nouvel ensemble d'exemples négatifs pour chaque formule générée. Que ce soit DL-Learner ou ces deux méthodes, elles sont assujetties à la possibilité de trouver de tels exemples négatifs. Pour cela, ces méthodes génèrent les exemples selon une supposition de monde fermé. C'est-à-dire qu'elles se basent sur l'hypothèse que toute donnée présente est vraie, et toute donnée absente est fausse. Elles ne sont donc pas compatibles avec les bases de connaissances basées sur l'hypothèse de monde ouvert qui suppose l'existence de données vraies mais pas encore enregistrées.

1.4 Conclusion

Ce chapitre survole la discipline de l'extraction de connaissance. Dans un premier temps on commence par définir la notion de *connaissance* comme un ensemble de régularités intéressantes dans un ensemble de données. Ensuite, on présente différents formalismes permettant d'exprimer cette connaissance, notamment les *logiques de description*. Finalement, on expose des méthodes pour extraire l'information, sous forme de règle d'association, servant d'étape intermédiare entre données et connaissance. Toutefois, on peut noter que les méthodes permettant d'extraire ces règles sur un jeu de données relationnelles sont limitées soit par des contraintes de format (la conclusion de règle peut être limitée à un littéral d'arité 2) ou d'annotation des données (exemple positifs et négatifs à fournir en entrée). Cette thèse propose donc l'analyse relationnelle de concept (ARC) pour

concevoir un ensemble de règles sans ces limitations. Les chapitres 2 et 3 sont les préambules mathématiques permettant de proposer notre méthode d'extraction de règles d'association par ARC au chapitre 4. Les chapitres 5, 6 et 7 sont des cas d'application de cette méthode d'extraction de règles.

CHAPITRE II

ANALYSE FORMELLE DE CONCEPTS

L'analyse formelle de concepts (AFC), en anglais *formal concept analysis*, est une méthode qui vise à extraire la structure conceptuelle d'un jeu de données unaire. Depuis le treillis complet servant de support à cette structure, on peut extraire une représentation de la base de règles d'association d'un jeu de données, notre sujet d'étude. Nous présentons donc ici la théorie de l'AFC en commençant par introduire la théorie des treillis présentée dans (Davey et Priestley, 2002; Garg, 2015). Puis, nous présentons les définitions fondamentales de l'AFC (Wille, 1982), avant de présenter les règles d'associations que cette méthode permet d'extraire.

2.1 Treillis

Dans cette section, nous introduisons d'abord les ensembles partiellement ordonnés, puis les treillis, qui en sont un cas particulier.

2.1.1 Ensemble partiellement ordonné

Une *relation binaire* R sur un ensemble X est un sous-ensemble quelconque $R \subseteq X \times X$. Pour $x, y \in X$, on écrit, en notation infixe, xRy pour indiquer que $(x, y) \in R$, et $x \not R y$ sinon. Une telle relation R est dite d'*ordre partiel (non strict)* si, et

seulement si, elle vérifie les propriétés suivantes :

réflexivité : on a xRx pour tout $x \in X$;

antisymétrie : pour tous $x, y \in X$, si xRy et yRx alors $x = y$;

transitivité : pour tous $x, y, z \in X$, si xRy et yRz alors xRz.

Si on remplace la propriété de réflexivité, par celle d'irréflexivité, c'est-à-dire $x \not R x$ pour tout $x \in X$ alors on la qualifie d'*ordre partiel strict*. De plus, que l'ordre d'une relation soit strict ou non, on le qualifie de *total*, et non de *partiel*, si et seulement si, pour tous $x, y \in X$, tels que $x \neq y$, on a xRy ou yRx. On dit alors que toute paire d'éléments est *comparable*.

Exemple 3. *L'inclusion définie sur l'ensemble des parties d'un ensemble* $A = \{0, 1, 2\}$ *est un ordre partiel. En effet, deux parties* $U, V \in \mathcal{P}(A)$ *ne sont pas nécessairement comparables deux à deux : si* $U = \{0, 1\}$ *et* $V = \{1, 2\}$*, ni* $U \subseteq V$ *ni* $V \subseteq U$ *ne sont vraies. À l'inverse, la relation usuelle* $\leqslant$ *sur* $\mathbb{R}$*, l'ensemble des nombres réels, est un ordre total.*

Un ensemble X est dit *partiellement ordonné* (en anglais, *poset*) s'il est muni d'une relation d'ordre partiel R. On le note (X, R).

Soit $(X, \leqslant)$ un ensemble partiellement ordonné et m_i (resp. M_i) $\in (X, \leqslant)$ tel que, quel que soit $x \in X$, on ait $x \nleqslant m_i$ (resp. $M_i \nleqslant x$). On appelle m_i (resp. M_i) un *élément minimal* (resp. *maximal*) de $(X, \leqslant)$.

Si un élément m vérifie que, pour tout $x \in X$, on a $m \leqslant x$, alors m est appelé *minimum* de $(X, \leqslant)$, en anglais *bottom*, et est noté $\perp$. Inversement, on note le *maximum*, en anglais *top*, du *poset* $\top$. Il vérifie que, quel que soit $x \in X$, on a $x \leqslant \top$.

Un *poset* peut être représenté par un graphe acyclique orienté. Dans une telle représentation, chaque élément du *poset* est un sommet du graphe, et l'ensemble

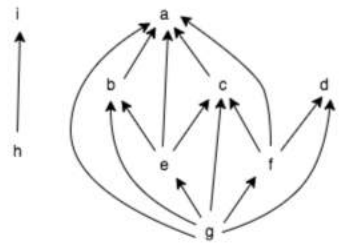


Figure 2.1: Exemple de *poset* $(X, <)$ représenté sous forme de graphe orienté.

des arcs décrit les couples constituant la relation d'ordre du *poset*. L'exemple 4 illustre une telle représentation. Néanmoins, une telle représentation, si on décide de la visualiser, comporte de l'information redondante, en raison de la propriété de transitivité de la relation d'ordre. On introduit donc la notion de diagramme de Hasse basée sur la notion de *précédence*, aussi appelée relation de *couverture*.

Exemple 4. *Soit l'ensemble* $X = \{a, b, c, d, e, f, g, h, i\}$. *La figure 2.1 dépeint le* poset (X, R), *où la relation d'ordre* R *est l'ensemble des couples* (u, v) *tel que* $u, v \in X$ *et qu'il existe un arc de* u *vers* v *dans le graphe. Ainsi, on a, entre autres,* hRi, gRc, fRa.

Soit $\leqslant$ une relation d'ordre et $<$ sa restriction irréflexive. La relation de couverture associée à $\leqslant$ et $<$, notée en infixe $\prec$, est définie, pour tous x et $y \in X$, par $x \prec y$ si et seulement si :

— $x < y$ et

— pour tout $z \in X$, $x < z \leqslant y$ implique $z = y$

Ainsi, $x \prec y$ signifie qu'il n'existe pas d'intermédiaire pour les relations d'ordre $\leqslant$ et $<$ entre x et y.

Le diagramme de Hasse du *poset* $(X, <)$ est le graphe $G = (X, \prec)$ dont les sommets sont les éléments du *poset* et les arcs sont les couples de la relation de précédence relative à la relation d'ordre du *poset*.

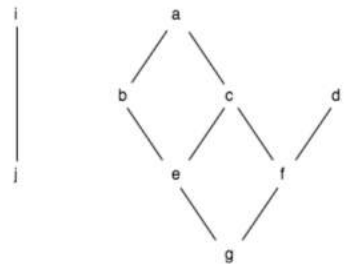


Figure 2.2: Diagramme de Hasse du *poset* de la figure 2.1

Les diagrammes de Hasse sont représentés verticalement, comme à la figure 2.2, où les plus petits éléments sont en bas et les plus grands en haut. Ainsi, il n'y a pas besoin de «flèche» pour représenter les arcs, des lignes pleines suffisent.

2.1.2 Treillis complet

Les treillis sont des *posets* présentant des contraintes particulières.

Soient deux *posets* $(Y, \leqslant_Y)$ et $(X, \leqslant_X)$ tels que $Y \subseteq X$ et $\leqslant_Y = \{(x, y) \in Y^2 \mid x \leqslant_X y\}$. On définit les notions de bornes inférieure et supérieure de Y, notées respectivement $\inf Y$ et $\sup Y$, comme suit. Soit $m \in X$. Si m vérifie que pour tout $y \in Y$, on a

— $m \leqslant y$ et

— pour tout $m' \in X$, $m' \leqslant y$ implique $m' \leqslant m$,

alors m est appelé *borne inférieure*, *infimum* de Y (en anglais *meet*). Elle est notée $\inf Y$. Dans le cas particulier où $|Y| = 2$, on note m par $a \wedge b$ ou $\inf\{a, b\}$, si $Y = \{a, b\}$.

La borne supérieure est définie de manière symétrique. Soit $s \in X$. Si s vérifie que pour tout $y \in Y$, on a

— $y \leqslant s$ et

— pour tout $s' \in X$, $y \leqslant s'$ implique $s \leqslant s'$

alors s est appelé *borne supérieure*, *suprémum* (en anglais *join*). Elle est notée sup. Si $Y = \{a, b\}$, on note s par $a \vee b$ ou $\sup\{a, b\}$.

Nous pouvons finalement introduire la notion de treillis et de treillis complet. Le *poset* $(X, \leqslant)$ est un treillis si et seulement si pour tous $x, y \in X$, $x \vee y$ et $x \wedge y$ existent. Il est qualifié de *complet* si et seulement si pour tout $Y \subseteq X$, $\sup(Y)$, et $\inf(Y)$ existent et $\sup(Y), \inf(Y) \in X$.

Exemple 5. *Considérons l'ensemble des partitions de l'ensemble* $\{1, 2, 3, 4\}$ *ainsi que la relation d'ordre* $\leqslant$ *suivante : pour deux partitions* X *et* Y*, on a* $X \leqslant Y$ *si et seulement si pour toute partie* $p \in X$*, il existe une partie* $q \in Y$ *telles que* $p \subseteq q$*. La figure 2.3*[1] *dépeint le diagramme de Hasse du treillis. Pour tout ensemble de partitions, le suprémum existe dans le treillis. Par exemple sup*$\{\{\{1\}, \{2, 3\}, \{4\}\}, \{\{1\}, \{2, 4\}, \{3\}\}, \{\{1\}, \{2\}, \{3, 4\}\}\} = \{\{1\}, \{2, 3, 4\}\}$.

Dans la prochaine section, on utilise la notion de treillis complet pour développer la théorie de l'AFC.

2.2 Analyse formelle de concepts

L'analyse formelle de concepts (AFC), introduite dans (Wille, 1982), analyse les jeux de données unaires. Ils sont composés d'objets, et d'attributs les caractérisant. C'est une méthode algébrique qui vise à découvrir les abstractions, appelés concepts formels, de tels jeux de données.

Dans cette section, nous présentons dans un premier temps les fondations mathématiques de l'AFC, notamment les notions de *contexte formel*, de *dérivation* et

1. figure réalisée par ed_g2s / CC BY-SA (http ://creativecommons. org/licenses/by-sa/3. 0/) disponible sur wikimedia commons free media repository

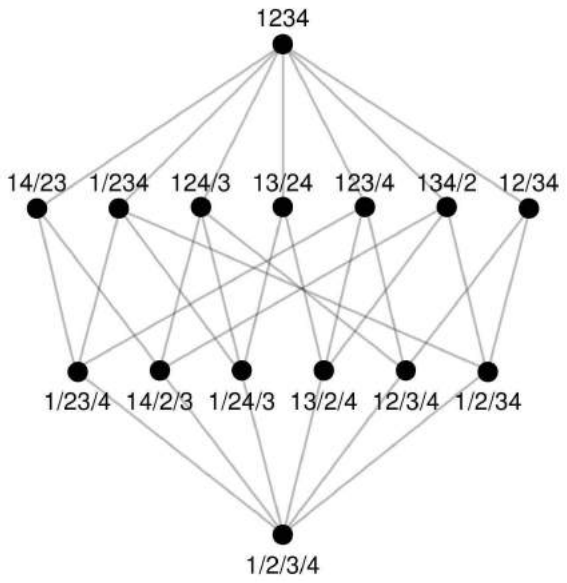


Figure 2.3: Treillis complet des partitions de l'ensemble $\{1, 2, 3, 4\}$

de *concept formel*. Puis, nous relions cette dernière notion au système de classes d'équivalences. Finalement, nous concluons cette section avec la notion de treillis de concepts.

Un *contexte formel* est un triplet $K = (O, A, I)$ où

— O est appelé ensemble des *objets*,

— A est appelé ensemble des *attributs* et

— $I \subseteq O \times A$ est une relation binaire, appelée la *relation d'incidence*.

Un *contexte formel* est une représentation d'un jeu de données unaires. Il peut être représenté par une table unaire. L'exemple 6 illustre cette définition ainsi que sa représentation.

Exemple 6. *Considérons quatre modèles d'automobile : la Twingo (abbréviée tw), la Tesla3 (t3), la Zoé (zo) et la Fiat 500 (f5). Parmi les diverses caratéristiques des véhicules, on retient les propriétés d'avoir un moteur électriques (el), d'avoir une puissance de minimum 100 chevaux (pu), d'être compacte et d'être plus petite que 4m (cp) et enfin de valoir moins de 20 000€ (nd). On peut définir le contexte*

K	el	pu	cp	nd
tw			$\times$	$\times$
$t3$	$\times$	$\times$		
zo	$\times$		$\times$	
$f5$		$\times$	$\times$	

Tableau 2.1: Contexte formel de l'exemple 6

formel $K = (O, A, I)$ *où* $O = \{tw, t3, zo, f5\}$ *est l'ensemble d'objets, et* $A = \{el, pu, cp, nd\}$ *l'ensemble d'attributs. Un tel contexte peut être représenté sous forme de table unaire, où les éléments de* O *correspondent aux lignes, les éléments de* A *aux colonnes, et la relation d'incidence* I *est identifiée par les croix. Une représentation est donnée dans le tableau 2.1. Une croix dans la case* $K[o_i, a_j]$ *indique que l'objet* o_i *possède l'attribut* a_j*.*

L'AFC vise à extraire les ensembles d'objets ayant des attributs communs. Nous commençons par présenter l'opération de base de l'AFC, appelée *dérivation*, avant de définir les *concepts formels*.

L'opération de *dérivation sur les objets* est une application de $\mathcal{P}(O)$ vers $\mathcal{P}(A)$. Pour un ensemble d'objets $X \in \mathcal{P}(O)$, sa dérivée, notée X', est donnée par

$$X' = \{a \in A \mid \forall\, o \in X,\ (o, a) \in I\} = \bigcap_{o \in X} \{a \in A \mid (o, a) \in I\} \tag{2.1}$$

De manière symétrique, l'opération de *dérivation sur les attributs* est une application de $\mathcal{P}(A)$ vers $\mathcal{P}(O)$. Pour un ensemble $Y \in \mathcal{P}(A)$, sa dérivée, notée Y', est donnée par

$$Y' = \{o \in O \mid \forall\, a \in Y,\ (o, a) \in I\} = \bigcap_{a \in Y} \{o \in O \mid (o, a) \in I\} \tag{2.2}$$

Ainsi, la dérivée d'un ensemble d'objets X est l'ensemble des attributs de A portés conjointement par tous les objets de X. De façon duale, la dérivée d'un ensemble d'attributs Y est l'ensemble des objets de O portant conjointement tous les attributs de Y. L'exemple 7 illustre ces définitions sur le contexte formel défini dans le tableau 2.1.

Les opérations de dérivation étant définies sur des ensembles différents et ayant des définitions symétriques, on se permet de parler uniquement de *dérivation*, sans préciser de manière explicite s'il s'agit de celle des objets ou des attributs.

Exemple 7. *Considérons le contexte formel défini dans l'exemple 6 et représenté dans le tableau 2.1. Alors, on trouve les dérivations suivantes :*

- $\{t3, f5\}' = \{pu\}$
- $\{zo, f5\}' = \{cp\}$
- $\{cp\}' = \{tw, zo, f5\}$
- $\{el, cp\}' = \{zo\}$

On note les propriétés suivantes de la dérivation, récapitulées dans les propositions 1 et 2, qui servent à démontrer certains éléments du chapitre 4

Proposition 1. *Soit un contexte $K = (O, A, I)$ et soient $X_1, X_2 \subseteq O$ et $Y_1, Y_2 \subseteq A$. On a $X_1 \subseteq X_2$ si et seulement si $X_2' \subseteq X_1'$. De plus, on a $Y_1 \subseteq Y_2$ si et seulement si $Y_2' \subseteq Y_1'$ (Ganter et Wille, 1999).*

Proposition 2. *Soit un contexte $K = (O, A, I)$ et soient $X \subseteq O$ et $Y \subseteq A$. On a $X''' = X'$ et $Y''' = Y'$ (Ganter et Wille, 1999).*

On appelle *concept formel* une paire $C = (X, Y) \in \mathcal{P}(O) \times \mathcal{P}(A)$ telle que $Y = X'$ et $X = Y'$. X est appelé l'*extension*, en anglais *extent*, et Y l'*intension*, en anglais *intent*, du concept C. Les concepts sont les abstractions fondamentales que l'AFC vise à extraire d'un contexte formel. L'exemple 8 illustre la notion en s'appuyant

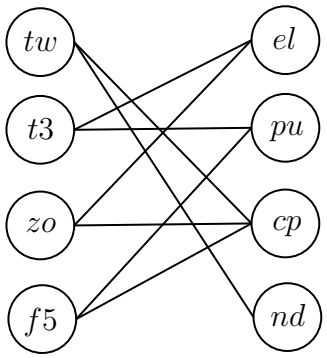


Figure 2.4: Représentation en graphe biparti du contexte K donné au tableau 2.1

sur le tableau 2.1. Intuitivement, un concept formel peut être visualisé comme un rectangle de croix maximal après permutation des lignes et des colonnes de la table.

On peut, alternativement représenter un contexte formel par un graphe biparti non orienté, c'est-à-dire un couple $G = (S, A)$, où S est l'ensemble des sommets et se décompose en deux parties P_1, P_2 telles que $S = P_1 \cup P_2$ et $P_1 \cap P_2 = \emptyset$, et où $A \subseteq P_1 \times P_2$ est appelé l'ensemble des arêtes. Dans une telle représentation, une première partie décrit les objets, la seconde partie les attributs, et les arêtes matérialisent la relation d'incidence. Les concepts sont alors exactement les bicliques maximales de ce graphe, c'est-à-dire les sous-graphes complètement connectés. Une représentation du contexte K (tableau 2.1) se trouve en figure 2.4. Toutefois, peu importe la représentation graphique préférée, l'intuition derrière la notion de concept formel est toujours la suivante : un concept est une paire d'ensemble d'objets et d'attributs, telle que

— les objets portent tous les attributs et sont les seuls du contexte à porter conjointement tous les attributs désignés ;

— chacun des objets porte tous les attributs, et aucun autre attribut du contexte n'est commun à tous les objets du concept.

Exemple 8. *Considérons le contexte formel de l'exemple 6 présenté au tableau*

2.1. Sur ce contexte, $(\{t3,\},\{pu\})$ *n'est pas un concept car* $\{pu\}' = \{t3, f5\}'$*. Toutefois, on a* $\{pu\}' = \{t3, f5\}'$ *et* $\{t3, f5\}' = \{pu\}$*, donc* $(\{t3, f5\}, \{pu\})$ *est un concept formel.*

Les concepts formels peuvent être caractérisés en tant que fermetures. L'opération de double dérivation $''$ est croissante et idempotente (Ganter et Wille, 1999), c'est-à-dire que pour tout $X \in \mathcal{P}(O)$ et $Y \in \mathcal{P}(A)$, on a $X \subseteq X''$, $(X'')'' = X''$, $Y \subseteq Y''$ et $(Y'')'' = Y''$. On l'appelle *fermeture* ou *clôture*. L'ensemble des concepts formels d'un contexte $K = (O, A, I)$ est donné par $\{(X'', X') \mid X \in \mathcal{P}(O)\}$ et de manière duale est aussi donné par $\{(Y', Y'') \mid Y \in \mathcal{P}(A)\}$ (Ganter et Wille, 1999). De plus, on peut noter que pour tout $Y \in \mathcal{P}(A)$, on a $Y' = Y'''$ et de manière duale pour tout $X \in \mathcal{P}(O)$, on a $X' = X'''$ (Ganter et Wille, 1999).

Exemple 9. *Continuons l'exemple 8 et considérons le contexte représenté au tableau 2.1. On a* $\{nd\}'' = \{tw\}' = \{cp, nd\}$*. Ainsi,* $\{nd\} \subseteq \{nd\}''$*. Et, comme* $\{cp, nd\}' = \{tw\}$*, la paire* $(\{tw\}', \{nd\}'')$ *est bien un concept formel.*

Les concepts formels sont intrinsèquement reliés à la notion de *classe d'équivalence*. Soit un ensemble $Y \in \mathcal{P}(A)$. On dit qu'un ensemble $Z \in \mathcal{P}(A)$ est *équivalent* à Y si et seulement si $Y' = Z'$. Ainsi, la *classe d'équivalence*[2] de Y, notée $[Y]$, est définie par $[Y] = \{Z \in \mathcal{P}(A) \mid Z' = Y'\}$. Elle vérifie également $[Y] = \{Z \in \mathcal{P}(A) \mid Z'' = Y''\}$ (Ganter et Wille, 1999). Et puisque la fermeture est une opération croissante, une classe d'équivalence $[Y]$ contient, pour la relation d'inclusion, un maximum unique, Y''. Le maximum unique d'une classe d'équivalence est appelé *fermé* ou *fermeture* (Ganter et Wille, 1999). Toutefois, elle contient un ou plusieurs éléments minimaux. Un élément $U \in [Y'']$ est dit minimal si pour tout $V \subset U$, $V \notin [Y'']$. Ces minimaux sont appelés *générateurs* de la classe. L'exemple 10 illustre ces

2. La notion de classe d'équivalence sur les objets est définie de manière duale mais on ne s'intéresse ici qu'à la définition pour les attributs, utilisée dans les règles d'association.

deux notions. Finalement, puisque les concepts formels sont décrits par l'ensemble $\{(Y', Y'') \mid Y \in \mathcal{P}(A)\}$, l'ensemble des fermés des classes d'équivalence correspond exactement à l'ensemble des intensions des concepts formels.

Exemple 10. *Dans le contexte représenté au tableau 2.1, on vérifie* $\{nd\}' = \{cp, nd\}'$. *Ainsi,* $\{nd\}$ *et* $\{cp, nd\}$ *appartiennent à une même classe d'équivalence. De plus, cette classe d'équivalence ne contient aucune autre combinaison d'attributs. Finalement, comme* $\{nd\} \subseteq \{cp, nd\}$, $\{nd\}$ *est un* générateur *et* $\{cp, nd\}$ *est le* fermé *de la classe d'équivalence.*

Il nous reste maintenant à introduire la notion de treillis de concepts. Soit $K = (O, A, I)$ un contexte formel. On note :

— $\mathcal{C}_K$ l'ensemble de tous les concepts formels de $\mathcal{P}(O) \times \mathcal{P}(A)$ et

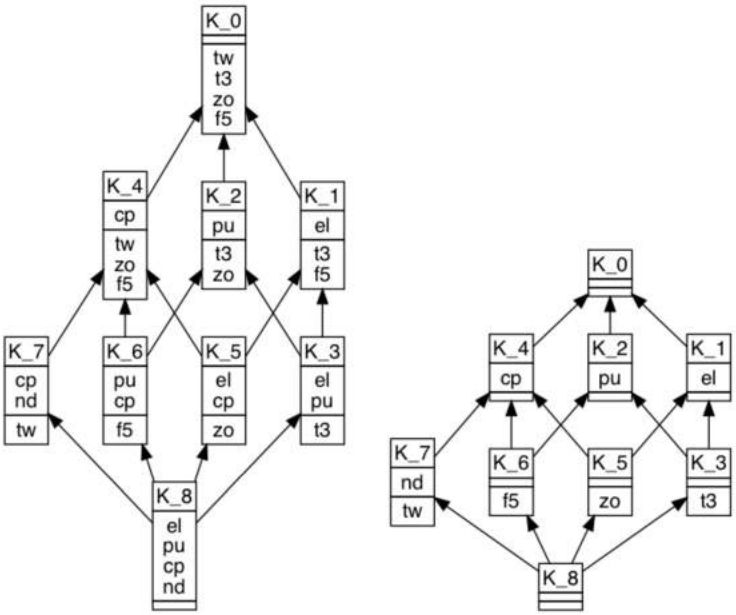
— $\leqslant_K$ la relation d'inclusion sur les extensions des concepts.

Le *poset* $\mathcal{L}_K = (\mathcal{C}_K, \leqslant_K)$ forme un treillis complet fini. On l'appelle **treillis de concepts** du contexte K (Ganter et Wille, 1999). De manière équivalente, la sélection de la relation d'inclusion inverse sur les intensions des concepts produit le même treillis de concepts (Ganter et Wille, 1999).

Ainsi, dans un treillis de concepts $\mathcal{L}_K = (\mathcal{C}_K, \leqslant_K)$, où $(X_1, Y_1) \in \mathcal{C}_K$ et $(X_2, Y_2) \in \mathcal{C}_K$, on a $(X_1, Y_1) \leqslant_K (X_2, Y_2)$ si et seulement si $X_1 \subseteq X_2$. L'exemple 11 illustre la notion de treillis complet et présente le format des figures de treillis de concepts utilisé dans le reste du manuscrit.

Il convient de noter qu'un treillis complet fini possède toujours un top et un bottom. En effet, puisqu'un treillis complet fini contient la borne inférieure et supérieure de tout sous-ensemble (non strict), il contient, entre autres, les bornes supérieures et inférieures de l'ensemble de tous les concepts.

Ces éléments *top* et *bottom* sont donnés par



(a) complet (b) compressé

Figure 2.5: Diagrammes de Hasse du treillis de concepts du contexte 2.1 [3]

— $\top = (O, O')$

— $\bot = (A', A)$.

Exemple 11. *La représentation du diagramme de Hasse de la table 2.1 se retrouve de manière exhaustive dans la figure 2.5(a). Chaque concept est représenté par une boîte de trois sections qui, de haut en bas sont le numéro d'identification du concept, son intension, son extension. Cette convention est utilisée dans tout le document*

On appelle *concept-objet* de o le concept (o'', o') et *concept-attribut* de a le concept (a', a''). Le *concept-objet* de o est la borne inférieure de l'ensemble des concepts contenant o dans leur extension ; et le *concept-attribut* de a est la borne supérieure de l'ensemble des concepts contenant a dans leur intension (Ganter et Wille, 1999). Cette proposition permet de réduire l'affichage d'un treillis de concepts. En effet,

les objets, comme les attributs, n'ont besoin que d'une seule mention dans le treillis si l'on considère l'ordre donné par l'inclusion. On parle alors de *représentation compressée*. L'exemple 12 illustre ce point.

Exemple 12. *Les figures 2.5(a) et 2.5(b) présentent toutes deux exactement l'ensemble des concepts du contexte présenté dans le tableau 2.1. Le treillis 2.5(b) est une représentation compressée du treillis 2.5(a). Chaque sommet numéroté identiquement sur les deux treillis présente exactement le même concept. La lecture d'un concept en représentation compressée est donc interprétée comme suit :*

- *l'intension d'un concept est l'ensemble des attributs des intensions représentées du concept considéré et de tout super-concept ;*
- *l'extension d'un concept est l'ensemble des objets des extensions représentées du concept considéré et de tout sous-concept.*

Voyons comment l'organisation des concepts, sous forme de diagramme de Hasse d'un treillis, permet l'extraction rapide de tous les fermés et les générateurs existant sur un contexte. Soit, pour un contexte K, deux concepts $C_1 = (X_1, Y_1)$ et $C_2 = (X_2, Y_2)$, tels que $C_1 \prec C_2$. On appelle *face* de C_1 relative à C_2 l'ensemble $F_{1;2} = Y_1 \setminus Y_2$ (Pfaltz et Taylor, 2002; Le Floc'h *et al.*, 2003). Soit $E = \{E_1, ..., E_n\}$ une famille d'ensembles. L'ensemble des *bloqueurs* de E est donné par $B_E = \{B_i \mid \forall E_j \in E, B_i \cap E_j \neq \emptyset\}$. Un bloqueur $B_i \in B_E$ est dit *minimal* si et seulement s'il n'existe pas $B_j \in B_E$ tel que $B_j \subseteq B_i$.

D'une part, l'ensemble des bloqueurs minimaux de l'ensemble des faces d'un concept décrit ses générateurs (Pfaltz et Taylor, 2002). Cette propriété a notamment été utilisée dans le développement d'un algorithme nommé JEN, qui extrait les générateurs des concepts efficacement grâce au diagramme de Hasse du treillis de concept, (Le Floc'h *et al.*, 2003). D'autre part, tout concept a pour intension un fermé de classe d'équivalence. Ainsi, l'organisation en diagramme de Hasse des

concepts d'un contexte permet d'obtenir immédiatement les fermés et d'extraire les générateurs de toutes les classes.

Une pléthore d'algorithmes ont été proposés pour le calcul des concepts d'un contexte : *In-Close* (Andrews, 2009), *Snow* (Szathmary *et al.*, 2008), *Charm* (Zaki et Hsiao, 2002), *Eclat* (Ogihara *et al.*, 1997), *A priori* (Agrawal *et al.*, 1993), *FP-Grow* (Han *et al.*, 2000)... La revue de littérature (Poelmans *et al.*, 2013) couvre une large partie de ces algorithmes. Certains d'entre eux permettent de calculer les concepts, et, directement, de les incorporer dans le treillis ; d'autres ne calculent que les concepts et nécessitent, pour concevoir le treillis à partir des concepts, un algorithme comme celui de *Nourine* (Nourine et Raynaud, 1999) ou *iPred* (Baixeries *et al.*, 2009).

Nous présentons ici un algorithme naïf mais didactique [4]. Il permet de comprendre la logique fondamentale derrière tous les autres algorithmes, qui en sont des versions optimisées. L'idée générale pour générer le diagramme de Hasse du treillis de concepts est la suivante : on initialise la procédure avec le *bottom* et ses parents directs, puis de manière récursive, on calcule et ajoute les autres concepts.

Initialisation

— création du noeud *bottom* du treillis

$$\mathcal{L}_K = (\mathcal{C}_K, \leqslant_K) = (\{(A', A)\}, \emptyset)$$

— intégration des parents directs de *bottom*.

  — on crée l'ensemble des concepts-objet $\mathcal{C}_1 = \{(o'', o') \mid o \in O\}$

  — puis on les ajoute au treillis comme parents de bottom :

4. Pour gagner en lisibilité et éviter une étape de récurrence dans l'initialisation, l'algorithme présenté ne considère pas les cas où l'ensemble d'attributs d'un objet est strictement inclut dans celui d'un second objet

$$\mathcal{C}_K \leftarrow \mathcal{C}_K \cup \mathcal{C}_1$$

$$\leqslant_K \leftarrow \leqslant_K \cup \bigcup_{c_i \in \mathcal{C}_1} (\perp, c_i)$$

Récurrence

Ensuite, on veut construire tous les autres concepts du treillis par récurrence. Pour cela, on effectue les opérations suivantes jusqu'à obtenir le treillis de concepts complet :

— prendre une paire de concepts existant encore jamais sélectionnée,

— calculer la borne supérieure de la paire et

— l'ajouter au treillis.

Formellement, on écrit

$$\forall x, y \in \mathcal{C}_K \,, \begin{cases} \mathcal{C}_K \leftarrow \mathcal{C}_K \cup (x \vee y) \\ \leqslant_K \leftarrow \leqslant_K \cup (x, x \vee y) \cup (y, x \vee y) \end{cases}$$

Pour $x = (x_E, x_I) \in \mathcal{C}_K$ et $y = (y_E, y_I) \in \mathcal{C}_K$ on calcule la borne supérieure par

$$(x \vee y) = (\mathrm{Int}', \mathrm{Int}) \text{ où } \mathrm{Int} = x_I \cap y_I$$

2.3 Règles d'association

La finalité de l'AFC, en terme d'extraction, est de trouver les règles d'association présentant les co-occurences entre les attributs du contexte. Une méthode naïve, sans AFC, serait de générer toutes les associations du contexte, et de calculer, pour chacune d'entre elles, les mesures désirées (telles que la confiance et le support), puis les ordonner selon ces mesures. L'ensemble des règles serait donc décrit par

$\{U \rightarrow V \mid \forall U, V \in \mathcal{P}(A)\}$.

Comme aucune contrainte n'est appliquée ni à U, ni à V, la cardinalité de cet ensemble est $|\mathcal{P}(A)| \times |\mathcal{P}(A)| = |\mathcal{P}(A)|^2 = (2^{|A|})^2 = 2^{2\times|A|}$. Toutefois, cet ensemble de règles, contenant toutes les combinaisons possibles, présente énormement de redondance. Par exemple, pour $U, V, W \in \mathcal{P}(A)$, si $U \rightarrow V$ et $U \rightarrow W$ sont de confiance 100% (et donc se comportent comme des implications mathématiques), alors $U \cup W \rightarrow V$ pourra aussi être présentée comme une règle de confiance 100%.

Il serait donc intéressant de trouver, pour un ensemble de règles d'association $\mathcal{AR}$, un sous-ensemble de règles $\mathcal{R} \subseteq \mathcal{AR}$ (le plus petit possible) qui, muni d'un système d'inférences, permet d'exprimer toutes les règles de $\mathcal{AR}$. Un tel ensemble $\mathcal{R}$ se nomme une *représentation* de $\mathcal{AR}$. Cette section présente tout d'abord différentes représentations possibles d'un ensemble de règles d'association avant de positionner l'AFC comme méthode support à l'extraction d'une représentation d'une base de règles.

2.3.1 Représentation

On ne s'intéresse ici, conformément à (Kryszkiewicz, 2002), qu'aux représentations qui sont :

- *Sans perte*, c'est-à-dire que toute règle de $\mathcal{AR}$ pourra être générée depuis $\mathcal{R}$ par le mécanisme d'inférence ;
- *Solide* c'est-à-dire que seules des règles de $\mathcal{AR}$ pourront être générées depuis $\mathcal{R}$ par le mécanisme d'inférence
- *Informatives*, c'est-à-dire que les règles de $\mathcal{R}$ présentent une confiance et un support, et que, par le mécanisme d'inférence, le support et la confiance des règles générées peuvent aussi être calculés.

Dans un premier temps, nous présentons différents mécanismes d'inférence. En-

suite, nous exposons les différentes représentations qui, munies d'un ou plusieurs mécanismes d'inférences, vérifient les trois propriétés sus-mentionnées. Enfin, nous justifions pourquoi l'AFC, grâce à l'organisation des concepts dans un treillis, est une excellente candidate à la production d'une telle représentation.

Commençons donc par présenter les mécanismes d'inférence :

Axiomes d'Armstrong (Armstrong *et al.*, 2002) On retient deux axiomes qui s'appliquent aux règles d'association de confiance 100% mais ne renseignent pas sur le support

(1) Si $\mathit{confiance}(X \to Y) = 1$ alors $\mathit{confiance}(X \cup Z \to Y) = 1$.
(2) Si $\mathit{confiance}(X \to Y) = 1$ et $\mathit{confiance}(Y \cup Z \to W) = 1$ alors $\mathit{confiance}(X \cup Z \to W) = 1$.

Transitivité de confiance (CTP) (Luxenburger, 1991) Sur des règles d'association, exactes ou non, si $X \subset Y \subset Z$ on peut déterminer une règle $X \to Z \setminus X$ depuis $X \to Y \setminus X$ et $Y \to Z \setminus Y$ grâce aux règles suivantes :

(1) $\mathit{support}(X \to Z \setminus X) = \mathit{support}(Y \to Z \setminus Y)$.
(2) $\mathit{confiance}(X \to Z \setminus X) = \mathit{confiance}(Y \to Z \setminus Y) \times \mathit{confiance}(X \to Y \setminus X)$.

Couverture (Kryszkiewicz, 1998) On appelle la couverture d'une règle l'ensemble $C(X \to Y) = \{X \cup U \to V \mid U, V \subseteq Y \ \mathit{et}\ U \cap V = \emptyset \ \mathit{et}\ V \neq \emptyset\}$. On peut alors présenter les propriétés suivantes :

(1) Si $p \in C(r)$, alors $\mathit{support}(r) \leqslant \mathit{support}(p)$ et $\mathit{confiance}(r) \leqslant \mathit{confiance}(p)$.
(2) Si $r \in \mathcal{AR}, C(r) \subseteq \mathcal{AR}$.

Inférence des clôtures (Zaki, 2000) On peut également déduire des inférences grâce aux propriétés de la clôture présentée dans la section précédente. Soit $X \subset Y$ et $r = X \to Y \setminus X$ une règle de $\mathcal{AR}$. On a :

(1) $support(r) = support(Y'')$.

(2) $confiance(r) = support(Y'')/support(X'')$.

(3) $X'' \rightarrow Y'' \setminus X''$ est aussi une règle de $\mathcal{AR}$.

Maintenant que nous avons une idée des quatre mécanismes permettant de concevoir un nouvel ensemble de règles depuis un ensemble donné, voyons pour un ensemble de règles $\mathcal{AR}$, quel sous-ensemble sans perte, solide, informatif et muni d'un ou plusieurs de ces mécanismes d'inférence, permet de construire $\mathcal{AR}$.

Kryszkiewicz (Kryszkiewicz, 2002) soulève le fait que, parmi les ensembles valides pour répondre à de telles contraintes, on compte la combinaison $\mathcal{IB}$ et $\mathcal{RI}$ complétées par $\mathcal{GB}$.

Les trois premières formes exploitent la fermeture de la conclusion et la propriété de générateur en prémisse.

$\mathcal{GB}$ est l'ensemble des règles de $\mathcal{AR}$ telles que la prémisse soit un générateur de la conclusion qui, elle, est un fermé. Cet ensemble de règles ne contient que des règles exactes, et, associé à l'opération de couverture, il permet de découvrir toutes les règles exactes de $\mathcal{AR}$.
$\mathcal{IB}$ est l'ensemble des règles de $\mathcal{AR}$ telles que la prémisse sois un générateur et la conclusion un fermé. La fermeture du générateur doit être strictement incluse dans la conclusion. Toujours avec l'opération de couverture seule, (Kryszkiewicz, 2002) montre qu'on peut dériver toutes les règles non exactes de $\mathcal{AR}$.
$\mathcal{RI}$ est un sous-ensemble de $\mathcal{IB}$ qui rajoute la contrainte selon laquelle la clôture de la prémisse doit être le plus grand fermé strictement inclus dans la conclusion. (Kryszkiewicz, 2002) montre qu'avec la couverture et la CTP on peut en dériver toutes les règles non exactes de $\mathcal{AR}$.

Exemple 13. *Illustrons ces trois représentations de l'ensemble des règles d'asso-*

ciation du contexte formel de l'exemple 6, présenté au tableau 2.1.

$\mathcal{GB}$: Considérons $[nd]$. On a $\{nd\}'' = \{nd, cp\}$ donc $\{nd, cp\}$ est le fermé de la classe $[nd]$. De plus, comme $\emptyset'' = \emptyset$ il n'existe pas de sous ensemble Y de $\{nd\}$ tel que $Y'' = \{nd, cp\}$. Donc $\{nd\}$ est un générateur de $\{nd, cp\}$ qui est un fermé. Ainsi $\{nd\} \rightarrow \{nd, cp\} \setminus \{nd\}$ est une règle de $\mathcal{GB}$.

$\mathcal{IB}$: Considérons $[cp]$. On a $\{cp\}'' = \{cp\}$ donc $\{cp\}$ est à la fois fermé et générateur de $[cp]$ (c'est donc une classe singleton). On trouve quatre fermés dans lesquels $\{cp\}''$ est strictement inclus : $\{cp, nd\}$, $\{pu, cp\}$, $\{el, cp\}$ et $\{el, pu, cp, nd\}$. Les règles $\{cp\} \rightarrow \{cp, nd\} \setminus \{cp\}$, $\{cp\} \rightarrow \{pu, cp\} \setminus \{cp\}$, $\{cp\} \rightarrow \{el, cp\} \setminus \{cp\}$ et $\{cp\} \rightarrow \{el, pu, cp, nd\} \setminus \{cp\}$ sont donc des règles de $\mathcal{IB}$

$\mathcal{RI}$: $\mathcal{RI}$ est un sous-ensemble de $\mathcal{IB}$. Parmi les règles présentées pour $\mathcal{IB}$, $\{cp\} \rightarrow \{el, pu, cp, nd\} \setminus \{cp\}$ n'est pas valide pour $\mathcal{RI}$. En effet, $\{cp\}''$ n'est pas le plus grand fermé inclus dans $\{el, pu, cp, nd\}$, puisque $\{cp\}'' \subset \{el, cp\} \subset \{el, pu, cp, nd\}$ et $\{el, cp\}$ est un fermé. Les trois autres règles sont bien dans $\mathcal{RI}$.

2.3.2 Base de représentation des règles produite par l'AFC

Cette sous-section détaille comment l'AFC procède pour extraire des règles d'association et rattache l'ensemble de règles produites à une représentation de type $\mathcal{RI} + \mathcal{GB}$ de la base de règles composée de toutes les combinaisons possibles du contexte formel.

La production des règles exactes, c'est-à-dire $\mathcal{ARE}$, est effectuée en étudiant chaque concept $C = (X_C, Y_C)$ et en matérialisant pour chacun de ses générateurs g_i, la règle $g_i \rightarrow Y_C \setminus g_i$. Le support d'une telle règle est donné par $|X_C|/|O|$ La production des règles approximatives, c'est-à-dire $\mathcal{ARA}$ est effectuée en étudiant chaque concept C et en matérialisant pour chacun de ses générateurs g_i et

chacun de ses sous-concepts direct $D_j = (X_{D_j}, Y_{D_j})$, la règle $g_i \rightarrow Y_{D_j} \setminus g_i$, le support d'une telle règle est donné par $|X_{D_j}|/|O|$ alors que sa confiance est donnée par $|X_{D_j}|/|X_C|$

Proposition 3. *L'ensemble des règles exactes $\mathcal{ARE}$ produites par AFC est exactement $\mathcal{GB}$, et l'ensemble des règles approximatives $\mathcal{ARA}$ est $\mathcal{RI}$.*

Démonstration. Les ensembles $\mathcal{GB}$ et $\mathcal{RI}$ sont définis ainsi (Kryszkiewicz, 2002)

$$\mathcal{GB} = \{X \rightarrow Y \setminus X \mid Y = Y'' \, et \, X \in gen(Y) \, et \, X \neq Y\}$$

$$\mathcal{RI} = \{X \rightarrow Y \setminus X \mid Y = Y'' \, et \, X'' \subset Y \, et \, \neg\exists Z, X'' \subset Z'' \subset Y et \, X \in gens\}$$

où $gen(Y)$ représente l'ensemble des générateurs de la classe $[Y]$ et $gens$ l'ensemble de tous les générateurs

Montrons d'abord que $\mathcal{ARE} = \mathcal{GB}$.
Tout d'abord, rappelons que l'intension d'un concept est le fermé de la classe des générateurs de ce même concept. On peut donc affirmer que toute règle $r \in \mathcal{ARE}$ vérifie les contraintes nécessaires pour vérifier $r \in \mathcal{GB}$. Ainsi, on a $\mathcal{ARE} \subseteq \mathcal{GB}$.

Montrons l'inclusion inverse. Soit $r \in \mathcal{GB}$. Par définition, la conclusion de r est un fermé. Il existe un concept calculé par AFC dont l'intension est ce fermé. Tous ses générateurs sont calculés, notamment le générateur en prémisse de r. Pour chacun d'entre eux, une règle dans $\mathcal{ARE}$ est produite, entre autres, r. Ainsi, si $r \in \mathcal{GB}$ alors $r \in \mathcal{ARE}$.
Finalement $\mathcal{GB} \subseteq \mathcal{ARE}$ et $\mathcal{ARE} \subseteq \mathcal{GB}$ nous permettent de vérifier que $\mathcal{ARE} = \mathcal{GB}$.

Montrons maintenant que $\mathcal{ARA} = \mathcal{RI}$

On montre tout d'abord que $\mathcal{ARA} \subseteq \mathcal{RI}$. Soit $r \in \mathcal{ARA}$, on peut l'écrire $g_i \rightarrow D_j.Intension \setminus g_i$. Comme D_j est un concept, et son intension un fermé, on a $D_j.Intension'' = D_j.Intension$. De plus, g_i est un générateur de C, donc $g_i'' = C.Intension$. Et, comme on a $D_j < C$, on a $C.Intension \subset D_j.Intension$. Ainsi, $g_i'' \subset D_j.Intension$. Finalement, s'il existait Z tel que $X'' \subset Z'' \subset Y$, alors il existerait un concept $B = (Z', Z'')$ tel que $D_j \subset B \subset C$. Cela contredirait l'hypothèse selon laquelle D_j est un sous-concept direct de C. Ainsi, $r \in \mathcal{RI}$. Finalement, nous avons $\mathcal{ARA} \subseteq \mathcal{RI}$.

Montrons maintenant $\mathcal{RI} \subseteq \mathcal{ARA}$. Soit $r \in \mathcal{RI}$, on peut écrire $r = X \rightarrow Y \setminus X$ avec les contraintes $Y = Y''\, et\, X \in gens\, et\, X'' \subset Y\, et\, \neg\exists Z, X'' \subset Z'' \subset Y$. Alors, puisque $Y = Y''$, il existe un concept $D = (Y', Y)$. De plus, on trouvera dans les concepts produits par AFC, le concept $C = (X', X'')$, dont X est un générateur. Sachant que $X'' \subset Y$, on peut déduire que $D \subset C$. De plus, comme $\neg\exists Z, X'' \subset Z'' \subset Y$, il n'existe donc pas de concept B tel que $D \subset B \subset C$. Ainsi, D est un sous-concept direct de C. Finalement, r vérifie toutes les contraintes pour justifier $r \in \mathcal{ARA}$.

En conclusion, on a bien $\mathcal{ARA} = \mathcal{RI}$, puisque $\mathcal{ARA} \subseteq \mathcal{RI}$ et $\mathcal{RI} \subseteq \mathcal{ARA}$.

$\square$

2.4 Conclusion

Dans ce chapitre, nous avons détaillé le formalisme des ensembles partiellement ordonnés, en particulier des treillis complets. Nous avons aussi présenté l'analyse formelle de concepts, une discipline supportant l'extraction de connaissance sur un jeu de données unaire appelé contexte formel. Elle vise à extraire toutes les abstractions, appelées concepts formels, de ces contextes et les organiser dans une hiérarchie ayant la forme d'un treillis complet.

Nous avons décrit comment cette hiérarchie, sous forme de treillis, soutient l'extraction de connaissance sous forme de règle d'association. Finalement, nous avons démontré que l'ensemble des règles extraites constitue une base de représentation de l'ensemble des règles possibles. Toutefois l'AFC ne permet de traiter que les jeux de données n'ayant qu'un type unique d'individu. Le chapitre 3 introduit l'analyse relationnelle de concepts, une extension de l'AFC aux jeux de données relationnelles.

CHAPITRE III

ANALYSE RELATIONNELLE DE CONCEPTS

L'analyse formelle de concepts (AFC), comme décrite au chapitre 2, vise à extraire la connaissance, sous forme de règles d'association, qui est distillée sur un jeu de données homogène. Le présent chapitre se concentre sur l'extension de cette méthode au paradigme multi-relationnel. Nos efforts se focalisent particulièrement sur l'analyse relationnelle de concepts (ARC), présentée dans (Rouane-Hacene *et al.*, 2013).

La première section présente succinctement les principales approches ayant été proposées pour étendre l'AFC au paradigme multi-relationnel. Ensuite, nous détaillons l'ARC dans sa version originale, et nous présentons les stratégies utilisées pour extraire la connaissance d'un jeu de données relationnelles en utilisant l'ARC.

3.1 Extensions relationnelles de l'AFC

Cette section survole des extensions de l'AFC au paradigme de la fouille de données multi-relationnelle.

Dans (Ferré et Ridoux, 2000; Ferré, 2006), Ferré propose d'étendre l'AFC pour qu'elle exploite, non plus des attributs, mais des combinaisons logiques d'attributs, qui permettent d'exprimer des relations entre les données. L'article décrit,

au travers d'une approche nommé *analyse logique de concepts* (ALC) une façon d'extraire des règles dont la prémisse et la conclusion sont des formules de la logique propositionnelle.

Une autre extension, nommée *Graph-FCA*, définie dans (Ferré, 2015), se base sur les graphes de connaissance, en anglais *knowledge graph*. Elle permet d'exploiter les relations d'arité quelconque. Cette extension définit un contexte comme un triplet $K = (O^*, A, I)$, liant des *groupes d'objets* à des attributs. Un concept, dans ce formalisme, a pour extension une relation entre objets, et pour intension un motif de graphe projeté (MGP), en anglais *projected graph pattern*. Ces MGP peuvent être interprétés comme des requêtes en SPARQL. Ils sont des ensembles d'arêtes du graphe de connaissance (ou des conjonctions de prédicats) dont les littéraux sont transformés en variables. Par exemple, supposons qu'une base de connaissance contienne les termes $pere(marie, gustave)$, $pere(anne, jean)$, qui affirment que Gustave et Jean sont les pères respectifs de Marie et Anne, ainsi que les littéraux $epouse(gustave, lea)$, $epouse(jean, marie)$, qui affirment que Léa est l'épouse de Gustave et Marie celle de Jean. Alors, on a $((x, y), \{pere(x, z), epouse(z, y)\}) \in A$ et les couples $(marie, lea), (anne, marie) \in O^*$ portent un tel attribut. Le calcul des concepts se fait à l'aide d'un algorithme qui prévoit des opérateurs pour vérifier l'inclusion et l'isomorphisme des MGP, et calcule l'infimum et le suprémum de deux MGP (Ferré et Cellier, 2016). Il est à noter que la technique de Graph-FCA, comme décrite par les même auteurs dans (Ferré et Cellier, 2019), ne considère que des n-uplets d'entités d'arité 1 (qui correspondent à des attributs dans l'AFC), ou 2 (comme les relations binaires dans ARC). Graph-FCA est donc compatible avec les formalismes avec des données en triplets (format du web sémantique), tel que le format Resource Descrption Framework (RDF), et des logiques de description, sur lesquels nous focalisons. Toutefois, Graph-FCA ne prend en considération que la présence ou l'absence de liens entre entités, ce qui correspond, en terme de

logique de description, à se limiter au quantificateur $\exists$.

Les *familles des puissances d'un contexte* (FPC), en anglais *power context family*, introduites dans (Wille, 1997) et détaillées, entre autres, dans (Prediger et Wille, 1999; Wille, 2002; Kötters, 2016), constituent une extension de l'AFC qui vise à exploiter les relations d'arités quelconques dans un jeu de données relationnelles. Les FPC sont constituées d'un ensemble de contextes $K^n = (O^n, A^n, I^n)$ où :

— $n = 1$, le contexte K^1 présente tous les objets, avec tous les attributs possibles. La relation d'incidence de ce contexte spécifie si un objet possède un attribut (tout comme dans la définition de contexte formel donnée au chapitre 2).

— $n > 1$, un élément $o \in O^n$ est un n-uplet ordonné d'objets, un attribut $a \in A^n$ est une relation d'arité n et I^n spécifie si les éléments d'un n-uplet o sont reliés par la relation a.

En combinant les concepts des treillis des différents contextes, et en reliant les éléments des concepts des contextes, on peut extraire les graphes conceptuels d'une FPC. Pour plus de détails sur ces objets, le lecteur peut se référer à (Sowa, 1983). Toutefois, à l'instar de Graph-FCA, les FPC se limitent à l'utilisation du quantificateur $\exists$ en logique de description.

L'ARC, introduite dans (Huchard *et al.*, 2002), vise à étendre l'AFC aux données relationnelles compatibles avec le modèle *entité-association* (Chen, 1976). Un tel modèle considère des relations binaires entre les objets. L'ARC enrichit la description des objets en intégrant des attributs dit *relationnels*, qui font apparaître les relations inter-objets. La section 3.2 est dédiée à sa description précise.

La capacité à appréhender ces combinaisons relationnelles a déjà motivé diverses applications de la méthode dans des domaines variés, tels que :

L'hydro-écologie. Dans les articles (Nica et al., 2016; Dolques *et al.*, 2016), ainsi

que la thèse (Nica, 2017), l'ARC est utilisée pour étudier les répercussions des variations des facteurs chimiques d'un cours d'eau, sur la composition biologique de celui-ci, et notamment l'impact sur sa faune.

La biologie. Alam et al. utilisent l'ARC pour faire le lien entre la présence de gènes spécifiques et les enzymes produites chez un patient (Alam *et al.*, 2013).

La détection de communautés. Guesmi et al. s'intéressent au fait de détecter des communautés d'auteurs (Guesmi *et al.*, 2016a; Guesmi *et al.*, 2016b). À l'aide des concepts formels et des attributs relationnels produits par ARC, ils détectent les groupes d'auteurs publiant sur des sujets similaires et participant aux mêmes conférences. Dans (Azmeh *et al.*, 2013), l'ARC est utilisée pour détecter des communautés d'actionnaires dans le but d'évaluer les priorités d'investissement.

La prise de décision industrielle. Dans (Wajnberg *et al.*, 2019a; Wajnberg *et al.*, 2019b), l'ARC permet de relier les caractéristiques de pièces usinées en aluminium à certains problèmes de production, permettant ainsi de prévoir les arrêts en amont et donc de réduire les coûts de production.

L'éducation. Dans (Encheva, 2015), l'ARC est utilisée pour relier les élèves à des styles d'enseignements, permettant ainsi la conception d'un système éducatif davantage personnalisé.

Le développement d'ontologies. L'ARC, combinée à des techniques de traitement automatique des langues naturelles, permet de concevoir une ontologie (cf sous-section 1.2.2 pour la définition d'ontologie) à partir d'un corpus de textes (Bendaoud *et al.*, 2007; Hacene *et al.*, 2008). De plus, De Maio et al. présentent un système d'annotations sémantiques transformant une phrase en proposition de la logique du premier ordre se basant sur l'ARC (De Maio *et al.*, 2014).

L'ingénierie logicielle, notamment le modèle orienté-objet, est le domaine ayant

motivé la création de l'ARC. En effet, par sa hiérarchie de concepts, elle est un outil de choix pour présenter une factorisation de classe ou une relation d'héritage. En particulier, elle peut être utilisée dans le contexte de la détection d'erreurs et la maintenance (Moha *et al.*, 2008; Miralles *et al.*, 2015; El Hamdouni *et al.*, 2010; Guédi *et al.*, 2013; Carbonnel *et al.*, 2015; Carbonnel *et al.*, 2018). Mais l'utilisation de la méthode dans le domaine est large puisqu'elle peut également servir à étudier la composition de services (Azmeh *et al.*, 2011; Mezni et Kbekbi, 2019) ou encore la détermination des clés de liage sur RDF (David *et al.*, 2018).

On trouve aussi des cas d'application de l'ARC à des **données séquentielles** qui étudient la relation entre symptômes et examen médical pour le diagnostic en tenant compte de la chronologie des observations ou l'impact de variation de la composition chimique d'un cours d'eau sur sa flore (Nica, 2017; Nica *et al.*, 2017; Nica *et al.*, 2016b). L'exemple 18 de la section 3.3 est issu de ces applications.

3.2 Formalisme de l'analyse relationnelle de concepts

L'analyse relationnelle de concepts (ARC) est une extension de l'AFC à des jeux de données relationnels. Elle intègre aux contextes, puis aux treillis, des *attributs relationnels* qui traduisent les relations entre les objets. La sous-section 3.2.1 définit le formalisme associé à de tels jeux de données relationnels. La sous-section 3.2.2 présente les *opérateurs de propositionnalisation* qui permettent de concevoir les *attributs relationnels* intégrés aux différents contextes. Cette intégration est introduite dans la sous-section 3.2.3. Finalement, la sous-section 3.2.4 détaille la procédure de calcul complète d'ARC.

3.2.1 Famille relationnelle de contextes

L'ARC est une méthode de la fouille de données multirelationnelles (FDMR). Elle exploite donc différents types d'objets ainsi que des liens entre ces derniers. Soient deux contextes formels $K_i = (O_i, A_i, I_i)$ et $K_j = (O_j, A_j, I_j)$. Une *relation* ayant pour domaine K_i (appelé aussi *source*) et pour co-domaine K_j (également appelé *cible*), est une relation binaire sous-ensemble de $O_i \times O_j$. On note $R_{i,j,k}$ la k-ième relation de domaine K_i et codomaine K_j.

L'AFC ne permet pas d'utiliser directement l'information comprise dans une telle relation. Via l'intégration d'un système de graduation, l'ARC permet d'étendre l'AFC pour intégrer cette information et ainsi caractériser les objets d'un contexte source en fonction *des relations entretenues avec d'autres objets*.

Une *famille relationnelle de contextes* (FRC) est une paire $(\mathcal{K}, \mathcal{R})$ telle que :

— $\mathcal{K}$ est un ensemble de contextes formels $K_i = (O_i, A_i, I_i)$

— $\mathcal{R}$ est un ensemble de relations $R_{i,j,k} \subseteq O_i \times O_j$ pour $i, j \in \{1, ..., |\mathcal{K}|\}$

L'exemple 14, inspiré de (Rouane-Hacene *et al.*, 2013), présente une telle famille relationnelle.

Exemple 14. *Prenons le contexte formel K_D de la table 3.1. Il présente des médicaments avec certains de leurs attributs tels que leurs principes actifs et leurs effets secondaires connus. Puis, munissons-nous d'un second contexte K_P, présenté en table 3.2. Les objets sont ici des patients, présentés avec leur tranche d'âge, leur sexe et les syndromes qu'ils présentent. Les treillis de concepts de ces deux contextes sont disponibles à la figure 3.1.*

Finalement, considérons la relation takes*, présentée au tableau 3.3, ayant pour source le contexte des patients, et pour cible celui des médicaments. Elle décrit, pour chaque patient, les médicaments qu'il prend. On se munit aussi de la re-*

Drugs	Doravirine	Efavirenz	Maraviroc	Raltegravir	Ritonavir	Tenofovir	BreathDisorder	Diarrhea	Fatigue	HairLoss	Headache	HeartFail	LiverDmg	Nausea	Rash	Vomit	StrangeDreams
Pilfetro	×	.	.	.	.	.	.	×	×	.	×	.	.	×	.	.	×
Isentress	.	.	.	×	.	.	.	×	.	.	×	.	.	×	.	.	.
Kaletra	.	.	.	.	×	.	.	×	×	.	×	.	×	×	×	×	.
Selzentry	.	.	×	.	.	.	.	.	.	.	.	×	.	.	×	.	.
Sustiva	.	×	.	.	.	.	.	×	×	.	.	.	.	.	×	×	×
Viread	.	.	.	.	.	×	.	×	.	.	.	.	×	×	.	×	.

Tableau 3.1: Contexte K_D. Décrit un ensemble de médicaments au travers d'attributs représentant leur molécule active et leurs effets secondaires connus

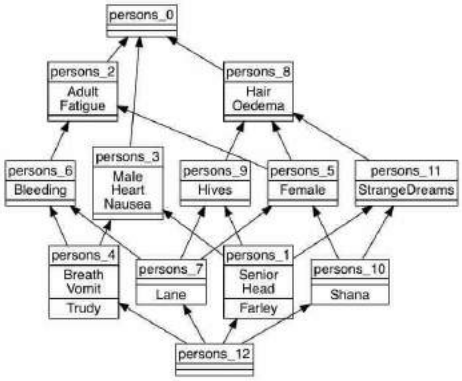
Patient	Senior	Adult	Male	Female	Bleeding	BreathDisorder	Fatigue	HairLoss	Headache	HeartFailure	Hives	Nausea	Oedema	Vomit	StrangeDreams
Farley	×	.	×	.	.	.	.	×	×	×	×	×	×	.	×
Lane	.	×	.	×	×	.	×	×	.	.	×	.	×	.	.
Shana	.	×	.	×	.	.	×	×	.	.	.	.	×	.	×
Trudy	.	×	×	.	×	×	×	.	.	×	.	×	.	×	.

Tableau 3.2: Contexte K_P. Décrit un ensemble de patients au travers d'attributs d'état civil et de symptômes médicaux présents

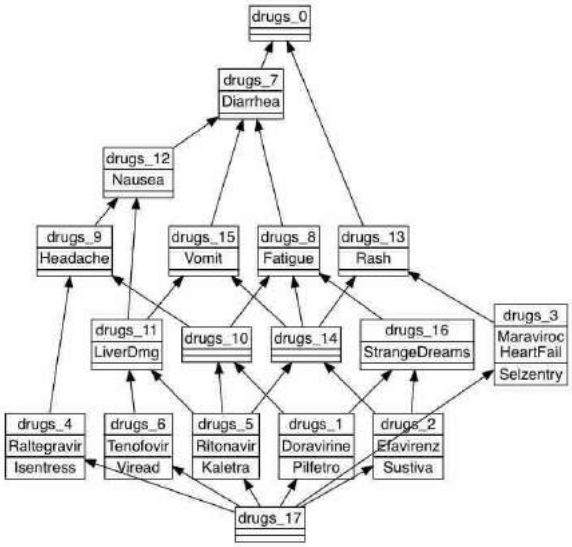
lation inverse, takes^{-1}*, correspondant à la transposée du tableau de* takes*. Elle décrit, pour chaque médicament, quels patients le prennent. On peut alors définir la famille relationnelle* $(\mathcal{K}, \mathcal{R})$ *telle que :*

— $\mathcal{K} = \{K_D, K_P\}$

— $\mathcal{R} = \{\text{takes}, \text{takes}^{-1}\}$



(a) Treillis $\mathcal{P}$: patients / caractéristiques et effets secondaires



(b) Treillis $\mathcal{D}$: médicaments / molécules et effets secondaires

Figure 3.1: Treillis de l'exemple 14

takes	Pilfetro	Isentress	Kaletra	Selzentry	Sustiva	Viread
Farley	.	.	×	.	.	×
Lane	.	.	.	×	×	.
Shana	.	.	×	.	.	×
Trudy	×	×	.	.	.	.

Tableau 3.3: Relation takes

takes^{-1}	Farley	Lane	Shana	Trudy
Pilfetro	.	.	.	×
Isentress	.	.	.	×
Kaletra	×	.	×	.
Selzentry	.	×	.	.
Sustiva	.	×	.	.
Viread	×	.	×	.

Tableau 3.4: Relation takes^{-1}

3.2.2 Opérateur de propositionnalisation

Lachiche définit la propositionnalisation comme un processus transformant explicitement un jeu de données relationnel en un jeu de données propositionnel (Lachiche, 2010), c'est-à-dire un ensemble de littéraux d'arité 1. Parmi les intérêts principaux de la propositionnalisation on compte la construction de caractéristiques, en anglais *features*, qui peuvent être combinées en hypothèses (Srinivasan *et al.*, 1996) et la réduction de l'espace de recherche (Lachiche, 2010). Il est à noter que ce biais de langage (le choix des opérateurs) ne garantit pas de pouvoir reconstituer l'information complète du jeu de données avec l'ensemble des énoncés émis par propositionnalisation (Krogel, 2005).

Dans l'ARC, on exploite les opérateurs de propositionnalisation issus des logiques de description. Ainsi, les intensions des concepts sont directement traduisibles dans ce formalisme. Toutefois, comme cette sous-section le présente, le choix des opérateurs n'est pas limité à ceux des logiques de description. Un opérateur de propositionnalisation permet de transformer une classe, via une relation en un attribut de la forme $\rho R.C$ où ρ est l'opérateur, R une relation et C le concept à transformer.

Parmi les opérateurs possibles on compte les opérateurs existentiel $\exists$, universel $\forall$, universel avec existence $\forall\exists$, de dénombrement $\leqslant_n$ et $\geqslant_n$, d'image partielle $\cap_{\leqslant p\%}$ et l'opérateur *tous* # que l'exemple 15 illustre.

Exemple 15. *Considérons une classe* Patient*, une classe* C = Médicament $\wedge$ Migraine $\wedge$ Nausée *décrivant l'ensemble des médicaments ayant pour effet secondaire de donner la migraine et des nausées. On considère la relation* prend *(*takes*) de l'exemple 14. On peut alors décrire grâce à un opérateur ρ les sous-ensembles de patients correspondant au concept* Patient $\wedge$ (ρ prend.(Médicament $\wedge$ Migraine $\wedge$ Nausée))

— Patient $\wedge$ $\exists$ prend.C *décrit l'ensemble des patients prenant au moins un médicament de la classe C.*

— Patient $\wedge$ $\forall$ prend.C *décrit l'ensemble des patients ne prenant que des médicaments de la classe C.*

— Patient $\wedge$ $\forall\exists$ prend.C *décrit la conjonction des deux opérateurs précédents.*

— Patient $\wedge$ $\leqslant_n$ prend.C *décrit l'ensemble des patients prenant au plus n médicaments de la classe C.*

— Patient $\wedge$ $\geqslant_n$ prend.C *décrit l'ensemble des patients prenant au moins n médicaments de la classe C.*

— Patient $\wedge$ $\cap_{\geqslant p\%}$ prend.C *décrit l'ensemble des patients prenant des médicaments dont au moins p% sont des médicaments de la classe C.*

— Patient $\wedge$ # prend.C *décrit l'ensemble des patients prenant au moins tous les médicaments de la classe C.*

3.2.3 Graduation

La *graduation* consiste à étendre un contexte $K_i = (O_i, A_i, I_i)$ d'une famille relationnelle de contextes en intégrant l'information relationnelle, sous forme d'attributs. Puisqu'un contexte est modifié par la graduation, on introduit les notations suivantes pour plus de clarté. On appelle $K_i^n = (O_i^n, A_i^n, I_i^n)$ le contexte K_i après n opérations de graduation. Le contexte dans sa forme initiale est noté $K_i^0 = (O_i^0, A_i^0, I_i^0)$. On note $\mathcal{L}_i^n$ le treillis de concept de K_i^n.

Les attributs relationnels traduisent les liens entre les objets de K_i et ceux de K_j par une relation $R_{i,j,k}$. Pour cela, la graduation utilise conjointement la relation $R_{i,j,k}$, les concepts du treillis $\mathcal{L}_j^q$ du contexte K_j^q et un opérateur de propositionnalisation ρ.

La graduation du contexte K_i^p est noté $K_i^+ = (O_i^+, A_i^+, I_i^+)$ où :

Nom	ρ	$Contrainte(\rho, r, o, C)$
Existentiel	$\exists$	$r(o) \cap Extension(C) \neq \emptyset$
Image partielle	$\cap_{\geqslant p\%}$	$\lvert r(o) \cap Extension(C) \rvert \geq p \times \lvert r(o) \rvert$
Universel large	$\forall$	$r(o) \subseteq Extension(C)$
Universel strict	$\forall\exists$	$r(o) \subseteq Extension(C)$ et $r(o) \neq \emptyset$
Tous large	$\#$	$Extension(C) \subseteq r(o)$
Tous strict	$\#\exists$	$Extension(C) \subseteq r(o)$ et $Extension(C) \neq \emptyset$

Tableau 3.5: Différents opérateurs de propositionnalisation

— $O_i^+ = O_i^0$: l'extension de contexte caractérise le même ensemble d'objets

— $A_i^+ = \{\rho\, R_{i,j,k} : C_{j,u} \mid C_{j,u} \in \mathcal{L}_j^q\}$: pour un opérateur ρ fixé, un nouvel attribut est généré pour chaque concept $C_{j,u}$ (u-ième concept du treillis cible $\mathcal{L}_j^q$). Un tel attribut est de la forme $\rho\, R_{i,j,k} : C_{j,u}$.

— $I_i^+ = \{(o, \rho\, R_{i,j,k} : C_{j,u}) \mid o \in O_i\,,\, C_{j,u} \in \mathcal{L}_j^q \,\text{et}\, Contrainte(\rho, R_{i,j,k}, o, C_{j,u})\}$: la relation d'incidence est complétée selon les contraintes apportées par l'opérateur ρ.

Le tableau 3.5 présente des opérateurs ainsi que leur sémantique. Basé sur le tableau 7 de définition des opérateurs de graduation de (Rouane-Hacene *et al.*, 2013), il ajoute l'opérateur d'image partielle $\cap_{\geq p}$. Un attribut relationnel utilisant cet opérateur met en avant qu'une partie de l'image d'un objet est contenue dans un concept cible. De plus, le tableau 3.5 réadapte les opérateurs de restriction de cardinalité quantifiée à une sémantique plus proche de celle des logiques de description présentées dans (Baader *et al.*, 2003).

L'exemple 16 illustre la graduation d'un contexte.

Exemple 16. *On considère ici la famille relationnelle présentée dans l'exemple 14. Pour graduer le contexte K_P^0, on utilise :*

— *l'opérateur de propositionnalisation* existentiel *: $\exists$.*

— *le treillis $\mathcal{L}_D^0$, disponible en figure 3.1b, associé au contexte K_D^0*

— *la relation* takes

Ainsi, K_P^+ *est donné par :*

- $O_P^+ = O_P$ *les objets restent Shana, Lane, Farley et Trudy*
- *Pour chacun des* 18 *concepts du treillis cible des médicaments (figure 3.1b) on crée un attribut (colonne) de la forme* $\exists takes^{-1} : C_{D,u}$ *où* u *est le numéro d'identification d'un concept, donc* $u \in [0; 17]$*. Ces colonnes forment* A_P^+*.*
- *La relation d'incidence considère l'opérateur* $\exists$ *et la relation* takes*. Un couple* $(o, \exists takes : C_{D,u}) \in O_P^+ \times A_P^+$ *est ajouté à la relation d'incidence* I_P^+ *si et seulement s'il existe dans l'extension du concept* $C_{D,u}$ *un objet* $\bar{o}$ *tel que* $(o, \bar{o}) \in$ takes*.*

Ainsi, si on prend la colonne $\exists takes : C_{D,13}$ *il y a des croix aux lignes de Farley et Shana, car ces patients prennent du Kaletra, un des médicaments de* $C_{D,13}$*. Lane, qui prend du Sustiva, valide aussi cet attribut. En effet, le Kaletra, comme le Sustiva font partie des objets du concept* 13*,* il existe au moins un élément de l'image de Shana, Farley et Lane, par la relation *takes*, contenu dans les objets du concept 13*. À l'opposé, Trudy prend uniquement du Pilfetro et de l'Isentress. Aucun de ces médicaments n'est présent dans le concept* 13*, donc Trudy ne porte pas l'attribut* $\exists takes : C_{D,13}$*. Les résultats se trouvent dans la table 3.6.*

3.2.4 Cadre algorithmique

L'ARC est un processus itératif qui intègre, dans chaque contexte, l'information relationnelle. L'algorithme 3 reprend la procédure de ARC définie dans (Rouane-Hacene *et al.*, 2013).

Les lignes 1 à 4 représentent l'initialisation : à chaque contexte de la famille relationnelle d'entrée, on applique la fonction CalculeTreillis qui retourne le treillis correspondant au contexte avec un algorithme développé par l'AFC (cf

persons	$\exists takes : C_{D,0}$	$\exists takes : C_{D,1}$	$\exists takes : C_{D,2}$	$\exists takes : C_{D,3}$	$\exists takes : C_{D,4}$	$\exists takes : C_{D,5}$	$\exists takes : C_{D,6}$	$\exists takes : C_{D,7}$	$\exists takes : C_{D,8}$	$\exists takes : C_{D,9}$	$\exists takes : C_{D,10}$	$\exists takes : C_{D,11}$	$\exists takes : C_{D,12}$	$\exists takes : C_{D,13}$	$\exists takes : C_{D,14}$	$\exists takes : C_{D,15}$	$\exists takes : C_{D,16}$	$\exists takes : C_{D,17}$
Farley	×	.	.	.	.	×	×	×	×	×	×	×	×	×	×	×	.	.
Lane	×	.	×	×	.	.	.	×	×	.	.	.	.	×	×	×	×	.
Shana	×	.	.	.	.	×	×	×	×	×	×	×	×	×	×	×	.	.
Trudy	×	×	.	.	×	.	.	×	×	×	×	.	×	.	.	.	×	.

Tableau 3.6: Contexte $K_{\mathcal{P}}^{+}$

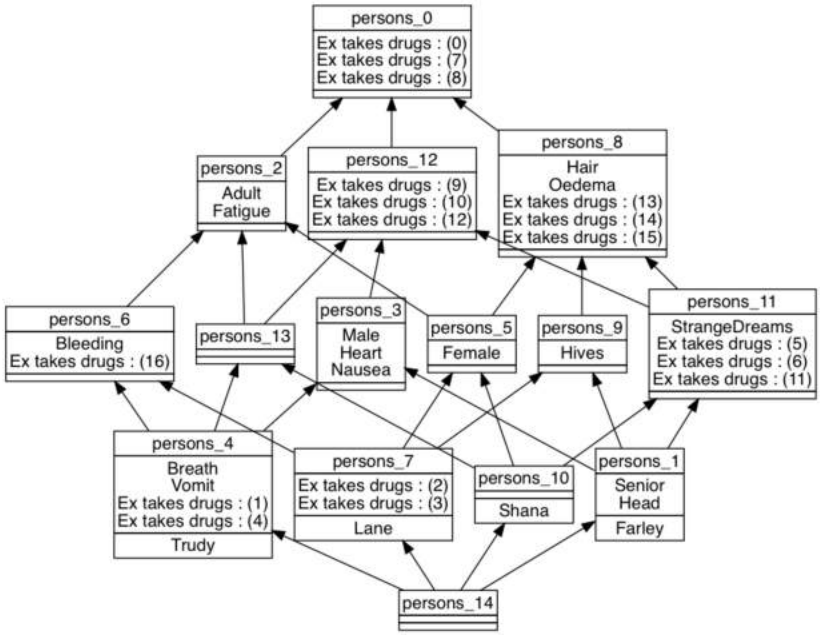


Figure 3.2: Treillis $\mathcal{P}$: Extension 1

Algorithme 3 : Procédure d'ARC

Données : Famille relationnelle de contextes
Résultat : Famille de treillis relationnels $\mathcal{L}$

1 $n \leftarrow 0$;
2 **pour** $K_i \in \mathcal{K}$ **faire**
3 | $\mathcal{L}_i^n \leftarrow$ CalculeTreillis(K_i)
4 **fin**
5 **tant que** *(¬ stable)* **faire**
6 | $n \leftarrow n + 1$;
7 | **pour** $K_i \in \mathcal{K}$ **faire**
8 | | **pour** $R_{i,j,k} \in \mathcal{R}$ **faire**
9 | | | $K_i \leftarrow K_i \cup$ Graduer($K_i, R_{i,j,k}, \rho, \mathcal{L}_j^{n-1}$)
10 | | **fin**
11 | **fin**
12 | **pour** $K_i \in \mathcal{K}$ **faire**
13 | | $\mathcal{L}_i^n \leftarrow$ CalculeTreillis(K_i)
14 | **fin**
15 | *stable* $\leftarrow$ `vrai`;
16 | **pour** $\mathcal{L}_i^n \in \mathcal{L}$ **faire**
17 | | *stable* $\leftarrow$ *stable* $\wedge$ estIsomorphe($\mathcal{L}_i^n, \mathcal{L}_i^{n-1}$)
18 | **fin**
19 **fin**
20 **retourner** $\mathcal{L}$

chapitre 2).

Les lignes 5 à 18 décrivent la procédure itérative de ARC :

— 1) On applique sur chaque contexte $K_i = (O_i, A_i, I_i)$ source d'une relation dans la famille relationnelle la fonction Graduer qui prend en entrée une relation, un opérateur de propositionnalisation, et le treillis cible de la relation et retourne la graduation du contexte K_i, c'est-à-dire $K_i^+ = (O_i^+, A_i^+, I_i^+)$(cf sous-section 3.2.3). On remplace alors le contexte $K_i = (O_i, A_i, I_i)$ par $K_i = (O_i, A_i \cup A_i^+, I_i \cup I_i^+)$.

— 2) On applique la fonction CalculeTreillis sur chaque contexte ainsi gradué. De nouveaux concepts peuvent émerger, comme le montre l'exemple

17. (lignes 12 à 14)

— 3) On applique enfin la fonction ESTISOMORPHE qui prend deux treillis et retourne la valeur booléenne `vraie` si et seulement si les diagramme de Hasse des treillis de concepts sont isomorphes, c'est-à-dire si aucun nouveau concept n'est généré. On considère alors que la famille relationnelle a atteint un état de *stabilité*, aussi appelé *point fixe* (lignes 15 à 18).

— 4) Si la famille relationnelle est stable, on retourne l'ensemble des treillis, achevant ainsi la procédure. Sinon, on procède à une nouvelle itération. (ligne 5 et 20)

Exemple 17. *Après graduation, le contexte K_P produit le treillis en figure 3.2. Ce «nouveau» treillis contient deux concepts de plus qu'à l'itération précédente (cf. concepts 12 et 13 sur la figure 3.1a)*

3.3 Interprétation des treillis relationnels

Dans la formulation actuelle de l'ARC, la production des treillis de concepts est réalisée. Toutefois, le treillis n'est pas une forme finale propice à l'extraction de connaissances. Notamment, une règle d'association présentant un attribut relationnel nécessite la consultation des treillis pour interpréter manuellement les concepts référés. Certes, il est possible d'étudier manuellement chaque concept pour déceler les caractéristiques des objets ; mais pour des treillis de grande taille, la tâche est ardue. Pour pallier cet écueil, différentes stratégies sont employées dans la littérature. Deux d'entre elles, que nous présentons dans cette section, sont toutefois prédominantes. Dans un premier temps, nous détaillons les schémas de clôture partiellement ordonnés (SCPO), en anglais *Closed Partially Ordered Patterns* (Nica *et al.*, 2016a; Nica et al., 2016), puis discutons des stratégies exploitant les composantes fortement connexes (Ferré et Cellier, 2018).

Le point commun de ces approches est de ne plus considérer les treillis séparément, mais de représenter l'ensemble des treillis sous la forme d'un multi-graphe orienté donné par la définition 3.

Définition 3. *On appelle* multigraphe au point fixe *d'une famille relationnelle un couple* $\mathcal{M} = (S, A)$ *dont*

- *l'ensemble des sommets* S *est composé de tous les concepts de tous les treillis de la famille relationnelle,*
- *l'ensemble* $A \subseteq S \times S$ *est composé d'arcs, étiquetés, qui peuvent représenter*
  - *la relation de subsomption entre deux concepts.*
  - *un attribut relationnel : si un concept* C_1 *porte l'attribut* $\rho r : C_2$ *dans son intension, on considère l'existence d'un arc* (C_1, C_2) *étiqueté* ρr*. Indifférement,* C_2 *peut appartenir au même treillis que* C_1*, ou à un autre treillis.*

Soit le multigraphe au point fixe $\mathcal{M} = (S, A)$ d'une famille relationnelle. Soit $C_u, C_v \in S$ deux concepts de $\mathcal{M}$.

Définition 4. *On dit que* $p = (C_1, C_2, \ldots, C_k)$ *est un* uv-chemin relationnel *de* G *si* $C_u = C_1$*,* $C_v = C_k$ *et* (C_i, C_{i+1})*, pour* $i = 1, 2, \ldots, k-1$*, est un arc formé par un attribut relationnel.*

Dans (Nica *et al.*, 2016a) et (Nica et al., 2016), on trouve le détail de l'extraction de schémas de clôture partiellement ordonnés (SCPO). La notion a été présentée dans ces articles uniquement pour des familles relationnelles intrinsèquement sans circuit, c'est-à-dire sans composante fortement connexe (CFC) sur le multigraphe. Pour chaque concept $C \in \mathcal{M}$ on définit le sous-graphe G_C formé par tous les nœuds C_t pour lesquels il existe un chemin relationnel de C à C_t. Parcourir en largeur G_C depuis C permet d'extraire un graphe orienté acyclique. En remontant

les arcs, des feuilles à la racine, on précise un attribut non relationnel et une relation dans chaque nœud. Si plusieurs relations (respectivement plusieurs attributs), doivent apparaître dans un même nœud, on remplace l'ensemble des relations (des attributs) par un point d'interrogation. Le résultat final est un SCPO. Une fois ces derniers extraits, on les ordonne dans une hiérarchie. L'exemple 18 illustre les notions.

Exemple 18. *Cet exemple, dont les figures, est adapté de (Nica* et al.*, 2016a) et traite de patients ayant la grippe A. On considère la famille relationnelle de contexte F comportant*

- *un contexte KVT décrivant des tests viraux, dont un des concepts est* `CKVT_6`*,*
- *un contexte KME décrivant des examens médicaux, contenant les concepts* `CKME_7` *et* `CKME_9`*,*
- *un contexte KS décrivant des symptômes, dont fièvre et toux, contenant les concepts* `CKS_2` *et* `CKS_3` *et*
- *les relations de précédence temporelle* ipb*, de découverte de symptômes aigüs* RhS *et modérés* RmS*.*

La figure 3.3 dépeint un extrait du multi-graphe de F après une exécution jusqu'au point fixe de l'ARC. Le concept `CKS_2` *contient le symptôme unique* fièvre *dans son intension et est uniquement lié à son prédécesseur par une relation* RmS*. Il est donc transformé en un nœud* fièvre modérée *dans le SCPO de la figure 3.4. Le concept* `CKS_3` *a, quant à lui, deux attributs dans son intension, traduits par un point d'interrogation. Comme ce concept n'est précédé que de la relation* RhS*, il est finalement traduit par* ?aigu *(dans la figure 3.4 par* ?high*). Ainsi, un tel SCPO peut être interprété par l'assertion suivante : avoir une fièvre modérée et un symptôme quelconque aigu est un signe précurseur à la grippe A.*

Une fois tous les SPCO extraits ceux-ci sont alors organisés dans une hiérarchie, du plus au moins spécifique, la figure 3.5 présentant un extrait. La hiérarchie des

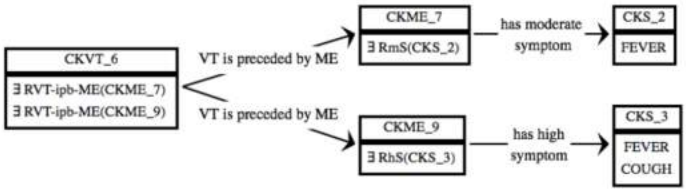


Figure 3.3: Extrait du multigraphe de la famille relationnelle de l'exemple 18 (Nica *et al.*, 2016a)

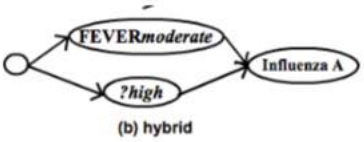


Figure 3.4: SCPO du concept `CKVT_6` de la figure 3.3 (Nica *et al.*, 2016a)

SPCO suit celle des treillis. En effet, on peut clairement voir la généralisation des SCPO dans un tel diagramme : par exemple, d'une part, le `C_KBIOS_720` *signale que* la présence d'une alerte verte quelconque et d'une alerte jaune sur la concentration de Phosphore, est corrélée à une alerte jaune sur l'indicateur IBGN *; d'autre part, le SCPO* `C_KBIOS_868` *indique que* la présence d'une alerte verte et d'une alerte jaune quelconques, est corrélée à une alerte jaune sur l'indicateur IBGN. *Clairement SCPO* `C_KBIOS_720` *est plus spécifique que SCPO* `C_KBIOS_868`.

Dans (Ferré et Cellier, 2018), les auteurs orientent leurs efforts sur les composantes fortement connexes (CFC) du multigraphe. Ils produisent la *condensation* du multi-graphe, c'est-à-dire calculent le graphe dont les CFC sont les nœuds et considèrent les arcs reliant ces CFC comme traduisant au moins un lien entre les deux CFC. Ils expliquent ensuite les composantes connexes de la plus générale à la plus spécialisée.

Exemple 19. *La figure 3.6, extraite de (Ferré et Cellier, 2018), présente un*

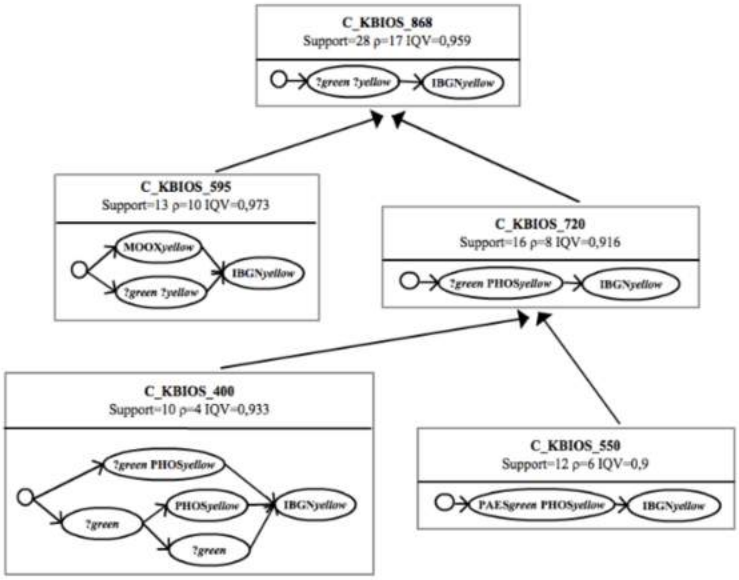


Figure 3.5: Hiérarchie des SCPO (Nica *et al.*, 2016a)

multigraphe composé d'un treillis de concepts décrivant des médicaments et d'un treillis des patients. Les liens pointillés traduisent la relation d'inclusion dans les treillis respectifs et les arcs les références par attribut relationnel quantifié par $\exists$*. La figure est décomposée en CFC, notées de G1 à G6, et les liens pleins désignent une relation de généralisation.*

G1 informe que tous les patients prennent des médicaments qui donnent la diarrhée et des médicaments qui provoquent de la fatigue. G2, G3, G4 sont des raffinements de ces déductions. Par exemple, G4 spécialise cette information aux patients ayant des saignements ; alors que G2 détaille d'une part, les patients ayant en plus une perte de cheveux et des oedèmes (ceux-là prennent des médicaments qui, en plus de fatigue et diarrhée, provoquent des éruptions cutanées et des vomissements), et d'autre part les médicaments (ces derniers sont tous pris par des patients ayant de l'urticaire et des patients qui sont de sexe féminin).

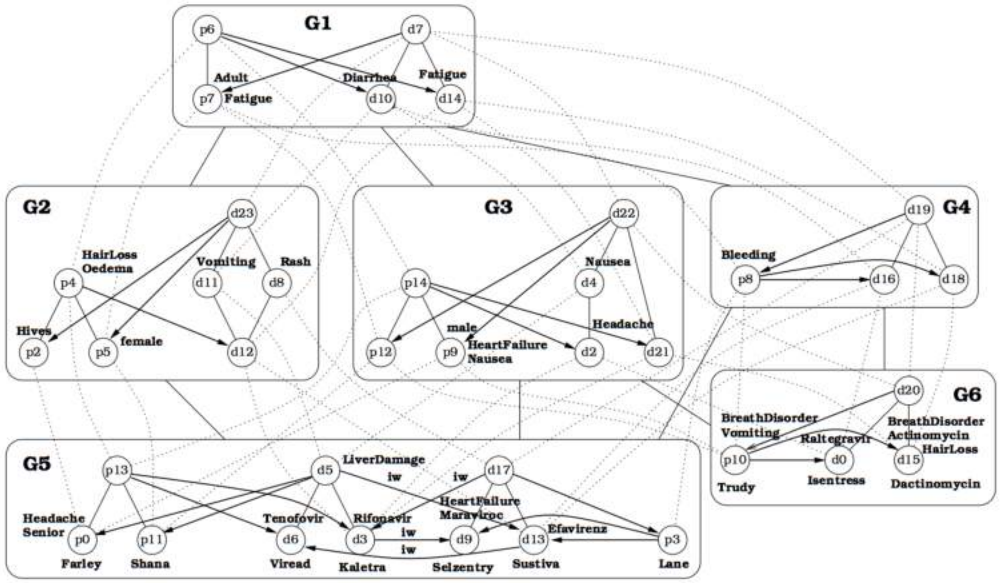


Figure 3.6: Composantes fortement connexes du multigraphe des treillis (Ferré et Cellier, 2018)

3.4 Conclusion

Dans ce chapitre, nous avons survolé différentes alternatives qui étendent l'AFC pour traiter des jeux de données relationnelles. L'ALC, Graph-FCA et les FPC se limitent à considérer la présence de liens entre les objets. Nous avons alors détaillé l'analyse relationnelle de concepts, une procédure qui intègre itérativement l'information relationnelle aux contextes formels exploités par AFC. Pour cela, l'ARC utilise des opérateurs de propositionnalisation appliqués à une relation binaire pour concevoir attributs relationnels décrivant les liens entre un objet et un concept. Cette approche permet de gagner en expressivité en quantifiant les liens entre objets et abstractions (les concepts formels).

Toutefois, les treillis résultant de l'ARC sont difficilement interprétables du fait de ces attributs relationnels. Les méthodes d'extraction de résultat se basent soit sur les descriptions en points fixes qui peuvent être récursives et dont l'interprétation

est peu aisée, soit sur des modèles intrinsèquement sans circuit, qui ne couvrent pas le cas général. Le chapitre 4 présente nos propositions pour extraire des règles d'association de ces treillis.

CHAPITRE IV

FONDEMENTS, LIMITES ET EXTENSIONS DE L'ANALYSE RELATIONNELLE DE CONCEPTS

Dans ce chapitre, nous approfondissons notre compréhension de l'ARC, telle que définie au chapitre 3. Nous en identifions les limites puis nous proposons des extensions de la théorie permettant de pallier ses lacunes. Tout d'abord, la section 4.1 vient préciser la notion de concept formel pour l'ARC. La section 4.2 détaille nos contributions permettant à l'ARC de réduire sa redondance et de contourner son aspect cyclique. La section 4.3 présente l'algorithme de l'ARC utilisant les formalismes et améliorations présentés. Finalement, la section 4.4 propose un ensemble de démonstrations mettant en évidence les avantages, sur un jeu de données relationnelles, de l'ARC par rapport à l'AFC.

4.1 Concept formel pour l'ARC

Selon la théorie présentée à la section 3.2, à chaque itération de l'algorithme, au moins un des contextes est étendu, en lui ajoutant de nouveaux attributs relationnels. Les définitions d'un concept et d'un attribut relationnel deviennent alors confuses : sur un contexte $K = (O, A, I)$ d'une famille relationnelle, une paire (X, Y) vérifiant $X' = Y$ et $Y' = X$ est qualifiée de concept formel ; or, comme l'ensemble A peut être étendu lors d'une itération, l'égalité $Y' = X$ peut

ne plus être vérifiée dans ce cas. Une telle paire peut-elle encore être décrite comme concept formel ? Si la réponse est non, que représentent les attributs relationnels vers ces «concepts incomplets» et que deviennent-ils ? S'ils sont supprimés, que faut-il faire des concepts portant un tel attribut, notamment si cet attribut est un générateur ? De telles suppressions peuvent enchaîner un effet en cascade. Nous nous attelons donc dans cette section à l'étude de ces questions en proposant une définition et en discutant des différentes alternatives.

Plusieurs possibilités s'offrent à nous pour redéfinir la notion de concept formel dans le paradigme de l'ARC et ainsi établir un traitement rigoureux des attributs relationnels. Nous discutons ici des trois formes de définitions qui nous semblent cohérentes avec la théorie de la littérature, détaillée dans la section précédente.

4.1.1 Concept formel évolutif

L'objectif principal de l'ARC est d'enrichir les descriptions des objets en intégrant l'information relationnelle comme partie intégrante de ces descriptions. En ce sens, il convient de privilégier l'extension pour décrire un concept. On propose donc les descriptions suivantes pour un contexte puis un concept formel dans l'ARC.

Définition 5. *Soit un contexte formel $K = (O, A, I)$. On appelle* suite de graduation *de K la suite de contexte (K_n) telle que $K_0 = K$ et, pour $K_n = (O, A_n, I_n)$ et son extension par graduation $K_n^+ = (O, A_n^+, I_n^+)$, on a $K_{n+1} = (O, A_n \cup A_n^+, I_n \cup I_n^+)$. On appelle* contexte de point fixe *d'une suite de graduation (K_n) le contexe K_p où p est l'itération à laquelle le processus d'ARC atteint la condition de stabilité, c'est-à-dire le point fixe.*

Pour éviter toute ambiguïté sur un contexte K_i d'une suite de graduation (K_n) la dérivation est notée ^{I_i}, la double dérivation ^{I_iI_i} etc.

Définition 6. *Soit* (K_n) *une suite de graduation de* $K = (O, A, I)$*. On appelle* concept formel évolutif *une paire* $C = (X, (Y_n))$ *où* X *est un ensemble d'objets et* (Y_n) *une suite d'ensemble d'attributs, tel que pour tous* n*,* $Y_n^{I_n} = X$ *et où* $X^{I_n} = Y_n$

Comme le présente la section 2.3 il reste à déterminer une représentation des règles qui soit minimale. Ces définitions nous permettent d'introduire les éléments nécessaires à la découverte de règles d'association sur ARC (cf section 4.2.1), qui reste l'objectif principal de notre processus d'extraction de connaissances. Toutefois d'autres alternatives sont envisageables pour définir les concepts formels. Nous en presentons deux dans les sous-sections 4.1.2 et 4.1.3.

4.1.2 Concept formel strict

La première alternative consiste à conserver la définition de concept comme établie dans l'AFC de manière stricte. Dans ce cas, une paire (X, Y) est un concept si et seulement si elle vérifie $X^{I_p} = Y$ et $Y^{I_p} = X$ pour la dérivation sur le contexte de point fixe K_p. Toutefois, l'ARC est un processus de découverte itérative : on ne connaît pas *a priori* les formes au point fixe des contextes. Ainsi, il est impossible de tester directement, à une itération quelconque si une paire forme un concept.

Dans ce cadre, la seule alternative est incrémentale : après l'étape du calcul des treillis de concepts à une itération $n + 1$, on doit déterminer les paires, concepts à l'étape n, qui conservent la propriété de concept formel, c'est-à-dire $X^{I_n} = Y$ et $Y^{I_n} = X$. Lorsqu'un tel couple $C = (X, Y)$ ne vérifie plus, à l'itération $n + 1$, les propriétés $X^{I_{n+1}} = Y$ et $Y^{I_{n+1}} = X$, il perd son «statut de concept». Alors, tout attribut relationnel $\rho r : C$ doit être réévalué ou supprimé. Une réévaluation consisterait à chercher à postériori des concepts de même extension. Choisir la suppression de manière systématique enclenche des pertes «en cascade» et conduit à un algorithme invalide comme le montre l'exemple 20.

Exemple 20. *Considérons la famille relationnelle utilisée au chapitre 3 dans l'exemple 14, traitant de médicaments et de patients. Comme tout patient prend au moins un médicament, tout concept du treillis des patients, après une étape de graduation, contient dans son intension l'attribut* $\exists takes : \top_D$*. De manière symétrique, tout médicament est pris par au moins un patient, ainsi, tout concept du treillis des médicaments contient* $\exists takes^{-1} : \top_P$*.*

Comme aucun des concepts des treillis de l'itération 0 *n'avait d'attribut relationnel, et qu'après une étape de graduation, tout concept a au moins un attribut relationnel, il vient qu'aucun concept à l'itération* 0 *n'est un concept après graduation. Finalement, tous les attributs relationnels calculés par graduation sont retirés. Enfin, les treillis sont isomorphes à ceux de l'itération* 0*. L'algorithme termine donc mais n'a pas intégré l'information relationnelle.*

La réévaluation consiste à modifier la définition d'attribut relationnel pour éviter les «pertes en cascade» de concepts. Lors de la phase de graduation, on propose, non pas de référer à un concept, mais à un ensemble d'objets. Dans un tel cas, puisque l'ensemble des objets de chaque contexte est constant avec les itérations, aucun attribut relationnel n'est invalidé par les itérations.

Exemple 21. *Considérons le concept 13 du treillis des médicaments à l'itération* 0 *:* $C_{D,13} = (\{\text{Kaletra, Sustiva, Salzentry}\}, \{\text{Rash}\})$*. Au lieu de créer par graduation, dans le contexte des personnes l'attribut* $\exists\text{takes} : C_{D,13}$*, on crée* $\exists\text{takes} : \{\text{Kaletra, Sustiva, Salzentry}\}$*.*

Proposition 4. *Soit la suite de graduation* (K_n) *d'un contexte* $K = (O, A, I)$*, et soit un ensemble d'objets* $X \subseteq O$ *tel que, il existe une itération* i *pour laquelle* $X^{I_i I_i} = X$ *sur* K_i*. Alors à toute itération* $j > i$*, on a* $X^{I_j I_j} = X$ *sur* K_j*.*

Démonstration. Soit $X \subseteq O$. Pour montrer que pour toute itération $j > i$, on

vérifie $X^{I_jI_j} = X$, il suffit de montrer que si $X^{I_iI_i} = X$ à une itération i, alors $X^{I_{i+1}I_{i+1}} = X$ à l'itération $i+1$. Soit (K_n) une suite de graduation d'un contexte $K = (O, A, I)$. On peut vérifier que $A_i \subseteq A_{i+1}$ et $I_i \subseteq I_{i+1}$.

Supposons $X = X^{I_iI_i}$. Par croissance de la double dérivation, on a $X \subseteq X^{I_{i+1}I_{i+1}}$. Montrons $X^{I_{i+1}I_{i+1}} \subseteq X$. Par construction, on a $X^{I_i} \subseteq X^{I_{i+1}}$, donc $X^{I_{i+1}I_{i+1}} \subseteq X^{I_iI_{i+1}}$. Il suffit donc de montrer que $X^{I_iI_{i+1}} \subseteq X^{I_iI_i}$. Posons $Y = X^{I_i}$, alors $Y \subseteq A_i$. Par construction, la relation d'incidence I_{i+1} préserve la dérivation I_i sur les attributs de A_i. Ainsi, puisque $Y \subseteq A_i$, on a $Y^{I_i} = Y^{I_{i+1}}$. Donc, on a $X^{I_iI_{i+1}} = X^{I_iI_i}$, ce qui amène $X^{I_{i+1}I_{i+1}} \subseteq X^{I_iI_i}$. Finalement, comme $X = X^{I_iI_i}$, on a $X^{I_{i+1}I_{i+1}} \subseteq X$. $\square$

Soit (K_n) une suite de graduation d'un contexte $K = (O, A, I)$ et K_p son contexte de point fixe et soit $X \subseteq O$ un ensemble d'objets. Alors, s'il existe $i \leqslant p$ tel que $C_i = (X, X^{I_i})$ est un concept formel sur K_i, par la proposition 4, $C_p = (X, X^{I_p})$ est un concept formel sur K_p. Ainsi, un attribut relationnel $\rho r : (X)$ conçu à l'itération $i+1$ est conservé sur A_p. Comme l'incidence des attributs relationnels n'est calculée que par rapport aux objets en relation et l'opérateur de graduation, les objets portant un attribut relationnel sont constants entre les itérations (cf exemple 22). Finalement, bien qu'on ne connaisse pas *a priori* les formes au point fixe des contextes et des treillis, on évite les destructions (potentiellement en cascade) des concepts en maintenant des attributs relationnels entre les itérations.

Exemple 22. *Considérons l'attribut* $a = \exists$takes : {Kaletra, Sustiva, Salzentry} *conçu à l'itération* 1 *de l'ARC. À cette première itération seuls les objets Lane, Shana et Farley portent l'attribut a, c'est-à-dire* $a^{I_1} =$ {Lane, Shana, Farley}. *Puisque l'attribut a est conservé dans tous les contextes jusqu'au point fixe, que la relation takes est constante entre toutes les itérations et que la sémantique de l'opérateur de propositionnalisation* $\exists$ *est invariable, on vérifie que pour tout*

$n \in [1; p]$, $a^{I_n} = \{\text{Lane, Shana, Farley}\}$

4.1.3 Accumulation

Une autre alternative est de conserver toutes les paires répertoriées par des concepts à une itération. Un concept est alors noté $C_{K_n,j}$ où K_n est le k-ième élément de la suite de graduation du contexte K (à l'itération n) et où j est son identificateur (numéro de concept). Un attribut relationnel a donc la forme $\rho r : C_{K_n,j}$

Bien que cette méthode propose une exhausivité plus importante que les autres alternatives présentées et évite toute suppression, elle présente un désavantage majeur : la création de redondance dans une méthode combinatoire déjà lourde en calculs. En effet, selon la proposition 4 le processus d'ARC crée une suite de concepts de même extension. De plus, on peut observer que pour les concepts $C_{K_n,j} = (X, Y_n)$ et $C_{K_{n+1},j} = (X, Y_{n+1})$, on a $Y_n \subseteq Y_{n+1}$. Ainsi, $C_{K_p,j}$ (pour K_p le contexte du point fixe de la suite (K_n)) contient toute l'information des versions antérieures. Ces duplications amènent donc de la redondance non pertinente.

Dans la suite du document, on considère les concepts selon la définition 6. Lorsqu'on décrit, sans précision, un concept comme une paire (X, Y) on réfère implicitement à Y comme étant l'intension au point fixe. De même, une opération de dérivation non indicée correspond à la dérivation sur le contexte au point fixe.

4.2 Améliorations de l'ARC

Cette section se consacre à confronter certaines limites de l'ARC en proposant des solutions théoriques. On étudie l'impact des attributs relationnels sur les règles d'association par ARC et la redondance de l'information.

4.2.1 Règles d'association

Comme présenté aux chapitres 1 et 2, le but ultime de notre procédure de construction de treillis est l'extraction de connaissance sous la forme de règles d'association. Qui plus est, le treillis de concepts sert de base d'extraction d'une représentation compacte de l'ensemble des règles d'association. Qu'un treillis soit produit dans un processus d'AFC ou d'ARC, comme on l'explique en section 2.3, la façon d'en extraire les règles est la même : pour chaque concept C on extrait toutes les règles $g \rightarrow Y \setminus g$ où g est un générateur de C, et Y est soit l'intension de C soit celle d'un concept qui est prédécesseur immédiat de C.

Toutefois, l'ARC introduit des attributs relationnels, qui, au même titre que les attributs non relationnels, sont incorporés aux règles. Or, pour qu'une règle soit utilisable, il faut pouvoir interpréter ces derniers, notamment en résolvant les références aux concepts de ces attributs. En effet, une règle de la forme $a_1, a_2 \rightarrow a_3, \rho r : C_{j,k}$ n'apporte aucune information si l'on ne peut pas caractériser $C_{j,k}$ Plusieurs options s'offrent donc à nous pour expliciter ces attributs relationnels :

Extension. Puisque l'extension des concepts est préservée tout au long des itérations, on peut remplacer la référence au concept par celle-ci. Dans un tel cas, un attribut relationnel caractérise alors les liens directs entre objets. Toutefois, une telle modélisation dénote sensiblement la nature même des règles d'association qui est de dégager des tendances, en se libérant des instances, au travers d'abstractions ; de plus, elle semble assez peu prometteuse lorsque la liste des objets d'un attribut relationnel est large.

Exemple 23. *Soient les attributs décrivant les effets secondaires de médicaments suivants, «vomissement» et «diarrhée». Soient les personnes Farley, Trudy et*

Lane. Et soit la relation takes^{-1} *qui décrit, pour un médicament donné, les patients qui l'utilisent dans un traitement.*

Considérons la règle

$$\text{vomissement, diarrhée} \rightarrow \exists \text{takes}^{-1} : (\text{Farley, Trudy, Lane})$$

Cette règle stipule que si un médicament donne des vomissements et de la diarrhée, alors il est pris par au moins une personne parmi Farley, Trudy et Lane. Plus la liste de personne est importante, plus l'information est diluée. Une telle règle est pertinente tant que la personne chargée de l'évaluer peut simultanément et manuellement consulter les dossiers des patients.

Intension À l'inverse, si on décide de remplacer la référence par l'intension du concept, plusieurs ambiguïtés émergent. Tout d'abord, il faut faire un choix sur la définition d'un concept, puisque celle-ci présente, comme décrit à la section 4.1, plusieurs options. En choisissant l'option du concept évolutif, donnée à la définition 6, plusieurs stades de l'intension sont possibles. En terme purement informationnel, pour un concept, la plus précise pour décrire ses objets est l'intension au point fixe. Toutefois, dans ce cadre, un problème inhérent à l'ARC émerge : les *dépendances cycliques*. En effet, les *chemins relationnels* du multigraphe peuvent former des circuits, comme le souligne la section 3.3. L'interprétation de tels circuits est un cas difficile, et nécessite d'utiliser la théorie des plus petits points fixes, comme le souligne (Baader *et al.*, 2003). L'exemple 24 illustre la présence des circuits.

Exemple 24. *Étudions les treillis aux figures 4.1 et 4.2. Nous pouvons observer que le concept* $persons_7$ *porte l'attribut* $\exists takes : drugs_{21}$ *et que le concept* $drugs_{21}$ *porte l'attribut* $\exists takes : persons_7$*. Comment alors résoudre les réferences de ces deux concepts ? De manière immédiate, on se rend compte que* $\exists takes : persons_7$

s'interprète par $\exists takes : persons\ (\exists takes : drugs_{21}, ...)$*, lui même détaillé par* $\exists takes : persons\ (\exists takes : drugs\ (\exists takes : persons_7, ...), ...)$ *etc..*

Certaines observations nous permettent cependant d'affirmer que cette circularité n'est pas inhérente aux *concepts* créés par l'ARC. Si on considère, pour un concept, l'intension à sa création, l'extraction de règles interprétables est alors possible puisqu'aucun circuit n'apparaît, comme le souligne le théorème 1, page 83.

Définition 7. *Soit le multigraphe au point fixe* $\mathcal{M} = (S, A)$ *d'une famille relationnelle. On appelle* multigraphe de création *le sous-graphe* $\mathcal{G} = (S, B)$ *où* $B \subseteq A$ *est l'ensemble des arcs formés uniquement par les attributs relationnels des intensions des concepts à l'itération de leur création.*

L'exemple 25 illustre la définition 7.

Exemple 25. *Considérons à nouveau la famille relationnelle utilisée au chapitre 3 dans l'exemple 14. Prenons le multigraphe au point fixe* $\mathcal{M}$ *de cette FRC. Il est donné par les graphes des treillis au point fixe du contexte des médicaments et du contexte des personnes, présentés par les figures 4.2 et 4.1, et des arcs représentant les attributs relationnels. Un extrait de ce graphe est présenté à la figure 4.3. Cet extrait correspond au sous-graphe de* $\mathcal{M}$ *limité aux concepts* P_0, P_2, P_6 *du treillis des personnes et aux concepts* $D_0, D_7, D_8, D_{16}, D_{19}, D_{20}$ *du treillis des médicaments ainsi que de tous les arcs sortant des sommets de ces 9 concepts. Pour des raisons de lisibilité, on marque les relations de subsomption par des flèches en lignes brisées, et on n'inclut dans la figure que les relations de subsomption entre parents directs. De plus, si pour deux sommets* S_1, S_2 *il existe à la fois l'arc* (S_1, S_2) *et* (S_2, S_1)*, on note cette paire d'arcs par une flèche à double orientation plutôt que deux arcs distincts.*

On peut noter que les concepts $P_0, P_2, P_6, D_0, D_7, D_8, D_{16}$ *sont créés à l'itération* 0*, sans aucun attribut relationnel, et que les concepts* D_{19}, D_{20} *sont créés à l'itération*

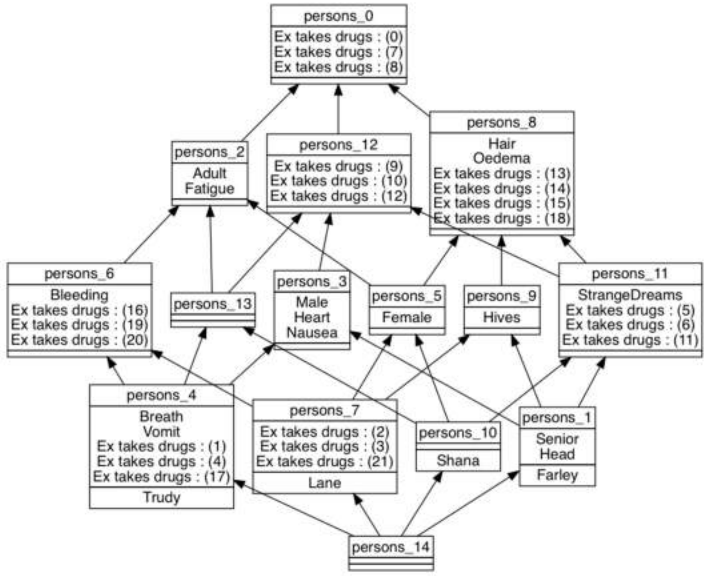


Figure 4.1: Treillis $\mathcal{P}$: point-fixe

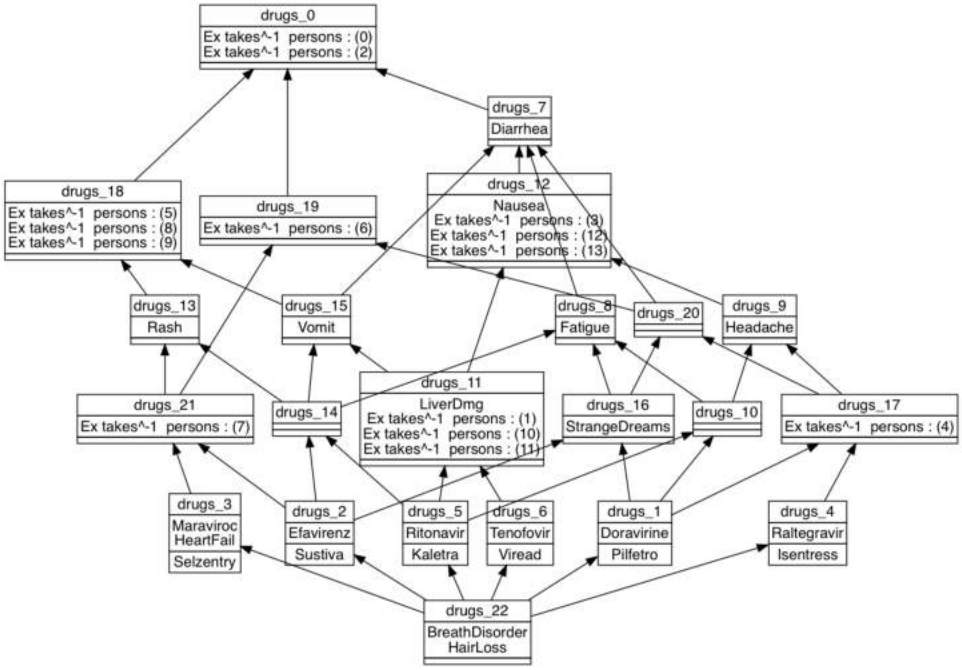


Figure 4.2: Treillis $\mathcal{D}$: point-fixe

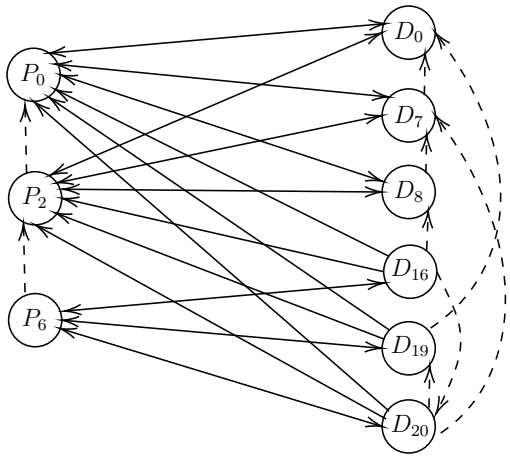


Figure 4.3: Extrait du multigraphe au point fixe de la FRC de l'exemple 25

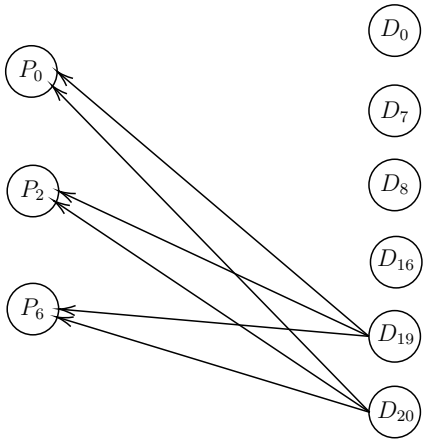


Figure 4.4: Extrait du multigraphe de création de l'exemple 25

1*, chacun avec les trois attributs* $\exists takes^{-1} : P_0, \exists takes^{-1} : P_2, \exists takes^{-1} : P_6$*. Finalement, la figure 4.4 dépeint l'intersection du multigraphe de création de la RCF et de la figure 4.3.*

Théorème 1. *Un multigraphe de création* $\mathcal{G}$ *est un graphe orienté sans circuit*

Démonstration. Montrons par récurrence, que tout sous-graphe $\mathcal{G}_n$ de $\mathcal{G}$ constitué des concepts existant à l'itération n est orienté acyclique. À l'itération 0, les concepts créés n'ont aucun attribut relationnel, donc il n'existe aucun chemin

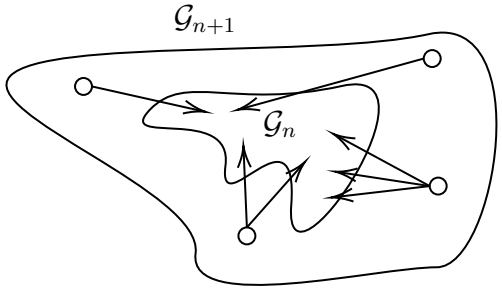


Figure 4.5: Le multigraphe $\mathcal{G}_{n+1}$ est composé des concepts existant à l'itération $n+1$ et, pour chacun de ces concepts, des attributs relationnels présents dans l'intension à l'itération de leur création. Il est composé du sous-multigraphe $\mathcal{G}_n$, son homologue à l'itération n, ainsi que des concepts générés à l'itération $n+1$ représentés par les cercles dans le bandeau $\mathcal{G}_{n+1} \setminus \mathcal{G}_n$. Les arcs représentent les attributs relationnels. Cette figure illustre le fait qu'il n'y a pas d'arcs entre ces nouveaux concepts et qu'il n'y a pas d'arc sortant de $\mathcal{G}_n$.

relationnel, donc la propriété est vraie.

Supposons maintenant une itération n pour laquelle la propriété est vraie, c'est-à-dire que $\mathcal{G}_n$ soit sans circuit. On illustre cette démonstration par la figure 4.5. Soit un concept C_{n+1} créé à l'itération $n+1$, c'est-à-dire $C_{n+1} \in \mathcal{G}_{n+1} \setminus \mathcal{G}_n$. Aucun attribut relationnel de $\mathcal{G}_{n+1}$ ne référence C_{n+1}, puisque les attributs relationnels référençant C_{n+1} sont produits par graduation à l'étape $n+2$. De plus, les attributs relationnels de C_{n+1} ne référencent que des concepts de $\mathcal{G}_n$. Finalement, comme $\mathcal{G}_n$ est orienté acyclique, C_{n+1} ne participe à aucun circuit sur $\mathcal{G}_{n+1}$. Ceci étant vrai pour tout concept de $\mathcal{G}_{n+1}$, $\mathcal{G}_{n+1}$ est graphe orienté acyclique.

□

Le théorème 1 permet donc d'interpréter les attributs relationnels de manière autonome, c'est-à-dire sans avoir besoin de consulter le multigraphe. Il suffit de remplacer la référence par l'intension à la création du concept. Toutefois, pour améliorer l'interprétabilité, on veut pouvoir réduire au besoin cet ensemble à un

ensemble d'attributs minimal décrivant de manière non ambiguë les objets d'un concept. Pour cela nous introduisons les propriétés suivantes sur les générateurs.

Lemme 1. *Soit* (K_n) *la suite de graduation d'un contexte* $K = (O, A, I)$ *et soit* $C = (X, (Y_n))$ *un concept créé à l'itération* c*. On note* p *l'itération de point fixe. Si* $g \subseteq A_c$ *est un générateur de* Y_c *sur le contexte* K_c*, alors pour tous* $i \in [c, p]$*,* g *est un générateur de* Y_i *sur* K_i*.*

Démonstration. Soit le concept $C = (X, (Y_n))$ créé à l'itération c. Soit $i \in [c, p]$ tel que $g \in A_c$ est un générateur de Y_i sur K_i. Montrons que g est un générateur de Y_{i+1} sur K_{i+1}. Pour cela on montre dans un premier temps que $g^{I_{i+1}I_{i+1}} = Y_{i+1}$ puis que g est minimal dans la classe d'équivalence de Y_{i+1} sur K_{i+1}.

(i) $g^{I_{i+1}I_{i+1}} = Y_{i+1}$

Par hypothèse, puisque g est un générateur de Y_i, on a $g^{I_i} = X$. Considérons maintenant Y_{i+1}. On vérifie que $Y_i \subseteq Y_{i+1}$. La graduation n'affectant pas la relation d'incidence sur des attributs d'itération antérieures, puisque $g^{I_i} = X$, on a $g^{I_{i+1}} = X$. Et, par définition du concept C, on a $X^{I_{i+1}} = Y_{i+1}$, et donc $g^{I_{i+1}I_{i+1}} = A_{i+1}$.

(ii) Minimalité

Supposons par l'absurde que g soit minimal à l'étape i, mais ne le soit pas à l'étape $i + 1$. Formellement, on pose les hypothèses suivantes : (1) sur K_i, on a $g^{I_iI_i} = Y_i$, $g \subseteq Y_i$ et il n'existe pas de $h \subset g$ tel que $h^{I_iI_i} = Y_i$; (2) sur K_{i+1} on a $g^{I_{i+1}I_{i+1}} = Y_{i+1}$, et il existe $q \subset g$ tel que $q^{I_{i+1}I_{i+1}} = Y_{i+1}$.

Puisque $q^{I_{i+1}I_{i+1}} = Y_{i+1}$, par la proposition 2 on a $q^{I_{i+1}} = X$. De plus, comme $q \subset g$ et $g \subseteq Y_i$, on a $q \subset Y_i$. Or, comme $q \subset Y_i$, par la proposition 1 on a $q^{I_i} = X$. Finalement, puisque par définition d'un concept on a $X^{I_i} = Y_i$, on trouve $q^{I_iI_i} = Y_i$, ce qui contredit l'hypothèse de minimalité de g à l'étape n.

Ainsi, des générateurs peuvent être ajoutés entre chaque itération, mais aucun générateur ne peut être supprimé entre deux itérations. Finalement, par induction, si un générateur g d'un concept C est présent à une itération n, il est présent à toute itération $m > n$.

□

Munissons-nous d'un ordre sur les attributs : un attribut a_n est inférieur à un attribut a_m si a_n est produit à une itération antérieure à la production de a_m . On assimile la présence des attributs standards lors de l'itération 0 à une «production». Si a_m et a_n sont produits lors d'une même itération, on les liste selon un ordre total : l'ordre lexicographique étendu de manière quelconque aux opérateurs de graduation

De cet ordre, on peut créer une valuation sur les attributs et ensemble d'attributs. Soit $K_p = (O, A_p, I_p)$ le contexte au point fixe d'un contexte K.

Définition 8. *Soit a le i-ème plus petit élément par l'ordre énoncé. On appelle la* valuation de l'attribut a, *notée $\nu(a)$, l'entier 2^{i-1}.*

Définition 9. *On appelle la* valuation de l'ensemble d'attribut $Y \subseteq A_p$ *la valeur dénotée par $\nu(Y) = \sum_{a \in Y} \nu(a)$.*

Des définitions précédentes on peut introduire la notion de canonicité.

Définition 10. *Soit G l'ensemble des générateurs d'une classe d'équivalence $[Y]$. On appelle un générateur g* canonique *pour la classe $[Y]$ si et seulement si $\nu(g) = \min_{u \in G}\{\nu(u)\}$.*

La canonicité est préservée tout au long du processus d'ARC

Lemme 2. *Si un générateur d'un concept C est canonique à une itération n il est un générateur canonique à toute itération $m > n$.*

Démonstration. Soit un concept $C = (X, (Y_n))$ et une itération i. Notons G_i l'ensemble des générateurs de Y_i et G_{i+1} l'ensemble des générateurs de Y_{i+1}. Par le lemme 1, on a $G_i \subseteq G_{i+1}$. Soit g un générateur canonique de G_i, montrons que g est aussi canonique sur G_{i+1}.

Par construction, $Y_i \subseteq Y_{i+1}$. Si $Y_i = Y_{i+1}$, on a $G_i = G_{i+1}$ et donc g est un générateur canonique de G_{i+1}. Sinon $Y_i \subset Y_{i+1}$. Comme $g \in G_i$, on a $g \subseteq Y_i$ et donc $\nu(g) \leqslant \sum_{a \in Y_i} \nu(a)$. Or, $\sum_{a \in Y_i} \nu(a) = 2^{|Y_i|} - 1$.

Soit $h \in G_{i+1}$. S'il existe $a \in h$ tel que $a \in Y_{i+1} \setminus Y_i$, comme par l'ordre donné $\nu(a) \geqslant 2^{|Y_i|}$, on a $\nu(h) > \nu(g)$. Ainsi, s'il existe $a \in h$ tel que $a \in Y_{i+1} \setminus Y_i$, h n'est pas canonique. Supposons maintenant que $h \subseteq Y_i$. Alors, on peut appliquer la dérivation sur K_i et $h^{I_i} = X$, donc $h^{I_i I_i} = Y_i$, c'est-à-dire h est dans la classe d'équivalence de Y_i sur K_i. Or, les minimaux de cette classe sont décrits par G_i donc il existe $\bar{g} \in G_i$ tel que $\bar{g} \subseteq h$, et $\nu(\bar{g}) \leqslant \nu(h)$. Finalement, comme g est le générateur canonique de G_i, $\nu(g) \leqslant \nu(h)$. Puisque quel que soit $h \in G_{i+1}$ on a $\nu(g) \leqslant \nu(h)$, on peut affirmer que g est un générateur canonique de G_{i+1}.

□

Lemme 3. *Soit un concept C généré à une itération i du processus d'ARC. Il existe un unique générateur canonique g de C à l'itération i et g est l'unique générateur canonique de C à toute itération $j > i$.*

Démonstration. Le générateur canonique est présent à toutes les itérations depuis la création du concept. En effet, à la création du concept, un ensemble de générateurs existe. L'ordre défini sur les attributs étant total, il existe un générateur canonique à la création du concept. La propriété 2 venant compléter cette propriété d'existence, on montre, par récurrence, que le générateur canonique est conservé de la création d'un concept jusqu'au point fixe.

□

Définition 11. *Soit le multigraphe au point fixe $\mathcal{M} = (S, A)$ d'une famille relationnelle. On appelle* multigraphe canonique *d'une famille relationnelle le sous-graphe $\mathcal{H} = (S, B)$ où $B \subseteq A$ est l'ensemble des arcs formés uniquement par les attributs relationnels du générateur canonique des concepts.*

Corollaire 1. *Soit $\mathcal{H}$ le multigraphe d'une famille relationnelle. $\mathcal{H}$ est un graphe orienté acyclique.*

Démonstration. Soit le multigraphe au point fixe $\mathcal{M}$ d'une famille relationnelle $\mathcal{F}$. Soient $\mathcal{G}$ le multigraphe de création et $\mathcal{H}$ le multigraphe canonique de $\mathcal{F}$. Par construction $\mathcal{H}$ est un sous graphe de $\mathcal{G}$. Or $\mathcal{G}$, par le théorème 1 est orienté acyclique, donc $\mathcal{H}$ aussi. □

Finalement, on peut considérer les générateurs canoniques pour représenter un concept dans un attribut relationnel. Une telle représentation, sans cycle, permet l'extraction de règles. Les chapitres 5, 6 et 7 illustrent l'utilité d'une telle représentation.

4.2.2 Raisonnement

Une autre limite notable de l'ARC à laquelle nous proposons une solution est ce qu'on peut appeler la *redondance relationnelle*. En effet, l'idée sous-jacente de la production de règles d'association par l'AFC, est d'extraire un ensemble minimal couvrant toute l'information. L'ARC, certes, utilise la méthode de l'AFC d'extraction de règles maximales dont les prémisses sont des générateurs et, de fait, extrait un ensemble de règles sans redondance en terme d'attribut. Toutefois, une règle d'association $A \rightarrow B$ où $A \cap B = \emptyset$ ayant des attributs relationnels,

peut présenter de la redondance qui provient de la définition des opérateurs de graduation, comme on peut le voir dans l'exemple 26.

Exemple 26. *Considérons le concept* $drugs_{18}$ *de la figure 4.2. L'attribut* $\exists \text{takes}^{-1}$: $person_5$ *est un générateur du concept. Parmi les règles d'implication on trouve* $\exists takes^{-1} : person_5 \rightarrow \exists takes^{-1} : person_8, \ldots$. *Or,* $person_5 \subseteq person_8$. *Et, par définition de l'inclusion, si* $\exists x \in person_5$ *alors* $x \in person_8$ *; ainsi, l'information* $\exists takes^{-1} : person_5 \rightarrow \exists takes^{-1} : person_8$ *n'apporte aucune information.*

Un moteur de raisonnement simple a donc pu être construit pour supprimer les attributs redondants d'un concept.

Le raisonneur va considérer un ordre sur les opérateurs de graduation. Du fait de nos cas d'applications, nous avons uniquement implémenté l'ordre concernant les opérateurs que nous avons utlisés.

Définition 12. *On définit l'ordre entre les opérateurs par la relation d'ordre, notée en infixe* $\lessdot$, *telle que pour* k *et* ℓ *éléments de* $]0;1[$ *avec* $k < \ell$ *on a :* $\exists \lessdot \cap_{\geq k} \lessdot \cap_{\geq \ell} \lessdot \forall\exists$.

De cet ordre sur les opérateurs, va être dérivé un ordre sur les attributs relationnels.

Définition 13. *Soient les attributs relationnels* $A = \rho_A r_A : C_A$ *et* $B = \rho_B r_B : C_B$. *On écrit alors que* $A \preccurlyeq B$ *(lire* A *est* moins informatif que B*) si et seulement si* $r_A = r_B$ *et l'une des deux propriétés suivante :*

- $\rho_A = \rho_B$ *et* $C_B \subseteq C_A$
- $C_A = C_B$ *et* $\rho_A \lessdot \rho_B$

Enfin, le raisonneur évalue chaque concept et ne conserve que les éléments maximaux parmi les attributs relationnels. Ainsi, le treillis de la figure 4.2, après

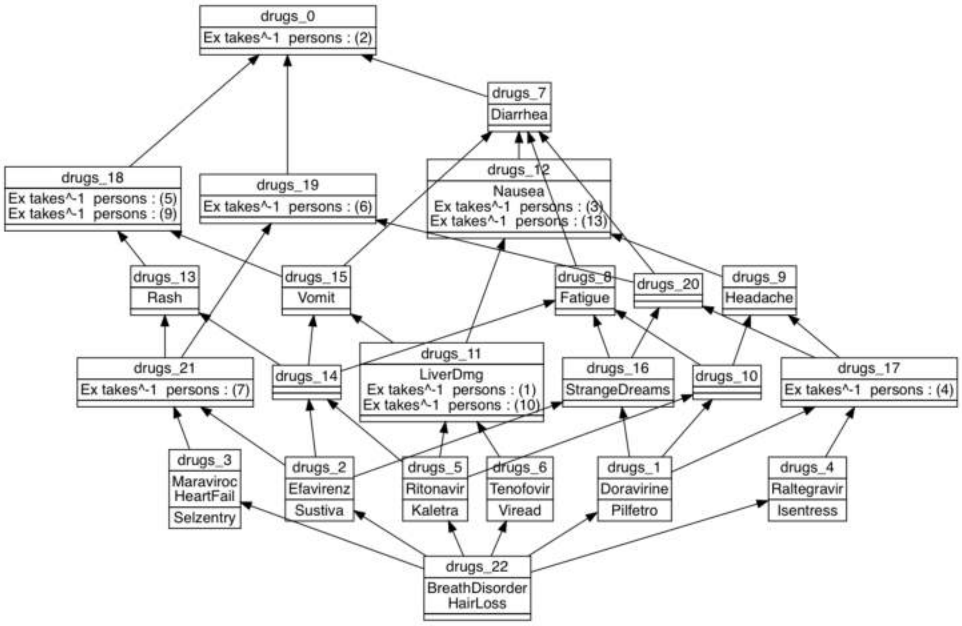


Figure 4.6: Treillis figure 4.2

traitement, correspond à celui en figure 4.6. Nous appelons cette opération la *compression relationnelle*.

4.3 Algorithme

Dans cette section on présente dans l'algorithme 4 une version modifiée de l'ARC.

Parmi les modifications proposées par rapport à l'algorithme 3, on note d'une part la possibilité de considérer plusieurs opérateurs de propositionnalisation par relation. Pour cela, on introduit le tableau associatif P qui, à une relation $R_{i,j,k} \in \mathcal{R}$ associe un ensemble d'opérateurs à appliquer pour la graduation (cf. ligne 10). De plus, en utilisant le lemme **??**, on peut se passer du test de stabilité par isomorphisme. À la place, un compteur sur le nombre global de concepts *nbConcepts* est introduit et la section itérative de l'algorithme s'arrête lorsque le nombre de concepts reste constant entre deux itérations successives (cf lignes 6 et

Algorithme 4 : Procédure d'ARC modifiée

Données : Famille relationnelle de contextes $(\mathcal{K}, \mathcal{R})$;
Tableau associatif P

Résultat : Famille de treillis relationnels $\mathcal{L}$

```
1  n ← 0;
2  nbConcepts ← 0;
3  pour K_i ∈ K faire
4  |  L_i^n ← CalculeTreillis(K_i)
5  fin
6  tant que nbConcepts ≠ Σ_i |L_i^n| faire
7  |  n ← n + 1;
8  |  nbConcepts ← Σ_i |L_i^{n-1}|;
9  |  pour R_{i,j,k} ∈ R faire
10 |  |  pour ρ ∈ P(R_{i,j,k}) faire
11 |  |  |  K_i ← K_i ∪ Graduer(K_i, R_{i,j,k}, ρ, L_j^{n-1})
12 |  |  fin
13 |  fin
14 |  pour K_i ∈ K faire
15 |  |  L_i^n ← CalculeTreillis(K_i);
16 |  fin
17 fin
18 pour L_i^n ∈ L faire
19 |  L_i^n ← Compresse(L_i^n);
20 |  L_i^n ← Canoniser(L_i^n);
21 fin
22 retourner L
```

8).

Pour se conformer à la définition 6 de concept formel, la fonction CALCULE-TREILLIS d'une part calcule les concepts nouvellement formés à l'itération n et complète les intensions des concepts existants (ligne 15). L'opération COMPRESSE (ligne 19) consiste à ne conserver que les attribut maximaux selon l'ordre $\preccurlyeq$ donné par la définition 13. Finalement, l'opération CANONISER (ligne 20) consiste à transformer les attributs relationnels pour référer aux générateurs canonique des concepts ciblés, conformément à la description proposée à la sous-section 4.2.1

4.4 Comparaison avec le paradigme non-relationnel

Maintenant que nous avons présenté et discuté le formalisme associé à l'analyse relationnelle de concepts, une question reste tout de même sans réponse : pourquoi utiliser une machinerie complexe et itérative du paradigme multi-relationnel plutôt que d'utiliser les techniques classiques de fouille de données ? Autrement dit, quels sont les avantages de l'ARC par rapport à l'AFC ? En effet, pour utiliser les techniques de fouille de données, on utilise souvent des techniques d'agrégation de tables telles que la jointure ou la semi-jointure (Spyropoulou et De Bie, 2011).

Dans les cas les plus triviaux, des réponses simples et évidentes semblent accessibles. Si plusieurs relations ont le même domaine et le même co-domaine, une jointure ou semi-jointure ne permet plus de faire la distinction entre les relations : on opère comme s'il n'existe qu'une seule relation, dont les éléments sont l'union de toutes les relations considérées, comme l'illuste l'exemple 27. Il est à noter que cet opération d'union se traduit exclusivement par l'opérateur de graduation *existentiel*. Puisque la semi-jointure semble porter des complications lorsque plusieurs relations sont prises en compte, on présente dans la sous-section 4.4.1, page 94, une comparaison avec l'ARC dans le cas d'une unique relation entre deux tables.

Exemple 27. *Soient un contexte de personnes, un autre de médicaments, et deux relations : l'une désignant si un patient est sous traitement d'un médicament, l'autre précisant si un patient est allergique à un médicament. En utilisant une jointure classique, un contexte a pour lignes des couples* $(personne, médicament)$ *et pour colonnes les attributs référant à ces couples. Aucune distinction entre les deux relations n'est perceptible. Supposons qu'une patiente,* Anne *prenne le médicament* Kaletra*, ayant pour effet secondaire de donner des* migraines *et de provoquer des* dommages au foie*, et qu'elle soit allergique au* Viread*, qui donne des* migraines *et provoque des* dommages au foie*. L'attribut* dommages au foie *est porté par les couples* (Anne, Kaletra) *et* (Anne, Viread) *sans qu'aucune distinction entre* allergie *et* prise *régulière ne soit mise en avant. Bien évidemment, artificiellement on peut pallier ce defaut en dupliquant, dans le contexte agrégé, les colonnes d'attributs précisant la relation valide : c'est-à-dire si les médicaments ont un attribut att on conçoit une colonne* prend-att *et une colonne* allergie-att*. Dans ce cas,* (Anne, Kaletra) *porte l'attribut* prend-dommage au foie *alors que* (Anne, Viread) *porte* allergie-dommage au foie

Une modélisation alternative qui contourne cet écueil consiste à concevoir un contexte agrégé qui prend en compte un opérateur de propositionnalisation. Un tel contexte a pour lignes les objets d'une première table, et pour colonnes, l'union des colonnes de la première et seconde table. L'incidence de l'objet vers les colonnes du premier contexte est conservée. La seconde partie, quant à elle, dépend de l'opérateur. Par exemple, si on choisit l'opérateur $\forall$, un objet o dans le contexte joint a un attribut du second contexte si et seulement si *tous* les objets du second contexte qui sont en relation avec o possèdent l'attribut considéré. La modélisation est décrite et illustrée en détail dans la sous-section 4.4.2, page 102, qui compare les résultats produits par AFC sur le contexte agrégé à ceux produits par l'ARC sur la famille relationnelle avant agrégation.

Dans cette section, nous nous attelons à démontrer que, pour les encodages les plus classiques des liens entre objets dans la constitution d'un contexte unique, pour tout concept $C = (X, Y)$ découvert par l'AFC on trouve par ARC un ensemble de concepts dont l'extension «égale» X et l'intension «approche» Y. Ainsi, le codage dynamique par ARC des liens entre objets par les attributs relationnels, d'une part, ramène le choix du type de jointure au choix des opérateurs de propositionnalisation, et, d'autre part, produit des résultats plus riches (ou, au pire cas, au moins aussi riches) que ceux de l'AFC sur un encodage en un seul contexte d'une famille relationnelle.

Les prochaines sous-sections constituent la présentation d'une modélisation, puis la démonstration de cette «inclusion» des résultats de l'AFC dans ceux de l'ARC.

Quel que soit le cadre de la démonstration, le raisonnement suit toujours les trois étapes suivantes :

1. déterminer la forme d'un concept formel C sur le contexte, utilisé par l'AFC, qui encode toute la famille relationnelle.
2. démontrer l'existence de concepts couvrant les mêmes objets dans la modélisation par ARC.
3. déterminer le rapport entre les intensions des concepts extraits des deux modélisations.

4.4.1 Opération de semi-jointure en cas mono-relationel

Prenons une RCF minimale, deux contextes et une relation, comme présentée dans la figure 4.7. On considère ici le cas concurrent où l'AFC est appliquée sur un contexte encodant la semi-jointure de cette RCF, comme présenté à la figure 4.8. Cet encodage consiste à créer les objets de $O_{\bowtie}$ comme les couples d'objets (o_1, o_2) où $o_1 \in O_1 \cup \{\perp\}$, $o_2 \in O_2 \cup \{\perp\}$, conformément à la modélisation de

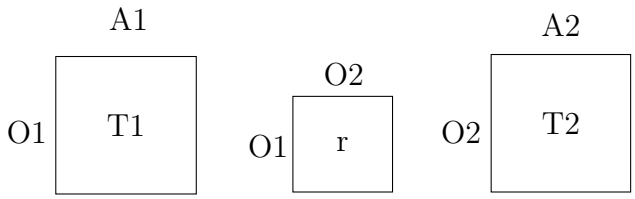


Figure 4.7: Modélisation ARC, deux tables K_1,K_2 et une relation r

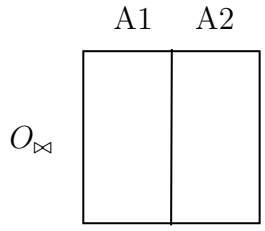


Figure 4.8: Modélisation après semi-jointure de la famille figure 4.7

O_1, O_2 en RCF figure 4.7. L'objet $\perp$ est un objet vide fictif sans attribut servant à compléter la semi-jointure. Trois cas se présentent pour définir les éléments de $O_{\bowtie}$:

— Si $o_1 \in O_1$, $o_2 \in O_2$, alors $(o_1, o_2) \in O_{\bowtie}$ si et seulement si $(o_1, o_2) \in r$

— Si $o_1 = \perp$, $o_2 \in O_2$, alors $(o_1, o_2) \in O_{\bowtie}$ si et seulement si $r^{-1}(o_2) = \emptyset$, c'est-à-dire s'il n'existe pas $x \in O_1$ tel que $(x, o_2) \in r$

— Si $o_1 \in O_1$, $o_2 = \perp$, alors $(o_1, o_2) \in O_{\bowtie}$ si et seulement si $r(o_1) = \emptyset$, c'est-à-dire s'il n'existe pas $x \in O_2$ tel que $(o_1, x) \in r$

L'exemple 28 illustre cette modélisation.

Exemple 28. *Considérons la famille relationnelle composée par le contexte de voitures de l'exemple 6 rapporté au tableau 4.1, ainsi qu'un contexte de personnes présenté au tableau 4.2. De plus on considère la relation de possession* pos *indiquant si une personne possède une voiture, présentée au tableau 4.3. Par jointure on obtient le contexte présenté au tableau 4.4.*

K_V	el	pu	cp	nd
tw			×	×
$t3$	×	×		
zo	×		×	
$f5$		×	×	

Tableau 4.1: Contexte des voitures

K_P	Senior	Adulte	Homme	Femme	Riche	Sportif
Fa	×		×			
La		×		×	×	
Sh		×		×		
Tr		×	×		×	×

Tableau 4.2: Contexte des personnes

pos	tw	$t3$	zo	$f5$
Fa	×			
La		×		×
Sh	×			×
Tr				

Tableau 4.3: Relation pos

$K_{\bowtie}$	Senior	Adulte	Homme	Femme	Riche	Sportif	el	pu	cp	nd
(Fa,tw)	×		×						×	×
(La,$t3$)		×		×	×		×	×		
(La,$f5$)		×		×	×			×	×	
(Sh,tw)		×		×					×	×
(Sh,$f5$)		×		×				×	×	
(Tr,$\perp$)		×	×		×	×				
($\perp$,zo)							×		×	

Tableau 4.4: Jointure de la famille relationnelle de l'exemple 28

Pour éviter toute ambiguïté, nous considérons les dérivations dans les contextes K_1 et K_2 toujours notées x', alors que la dérivation dans le contexte de jointure est notée x^∇ (et la double dérivation $x^{\nabla\nabla}$).

On s'intéresse dans un premier temps à décrire un concept formel du contexte de jointure. Soit un ensemble d'objets quelconque $X \subseteq O_\bowtie$. Donc, pour tous $(o_1, o_2) \in X$ on a $o_1 \in O_1 \cup \{\perp\}$ et $o_2 \in O_2 \cup \{\perp\}$. Ainsi, par définition, $C = (X^{\nabla\nabla}, X^\nabla)$ est un concept formel. Prenons à présent les projections sur les premiers et seconds éléments des couples de $X^{\nabla\nabla}$, c'est-à-dire $\pi_1 = \{o_1 \mid \exists o_2, (o_1, o_2) \in X^{\nabla\nabla}\}$ et $\pi_2 = \{o_2 \mid \exists o_1, (o_1, o_2) \in X^{\nabla\nabla}\}$. On commence par définir $X^{\nabla\nabla}$ en fonction de ces projections.

Lemme 4. *On a* $X^{\nabla\nabla} = (\pi_1 \times \pi_2) \cap O_\bowtie$

Démonstration. Soit $(u, v) \in X^{\nabla\nabla}$. Par définition $X^{\nabla\nabla} \subseteq O_\bowtie$ donc $(u, v) \in O_\bowtie$. De plus, par construction $u \in \pi_1$ et $v \in \pi_2$ donc $(u, v) \in \pi_1 \times \pi_2$. Ainsi, $X^{\nabla\nabla} \subseteq (\pi_1 \times \pi_2) \cap O_\bowtie$.

Soit $(u, v) \in (\pi_1 \times \pi_2) \cap O_\bowtie$. Puisque $(u, v) \in \pi_1 \times \pi_2$ il existe $\tilde{u}$ et $\tilde{v}$ tels que $(u, \tilde{u}) \in X^{\nabla\nabla}$ et $(\tilde{v}, v) \in X^{\nabla\nabla}$. Or par construction $\{(u, \tilde{u}), (\tilde{v}, v)\}^\nabla \subseteq u' \cup v'$. Et, puisque $(u, v) \in O_\bowtie$ on peut écrire $(u, v)^\nabla = u' \cup v'$. Ainsi, comme par propriété de dérivation on a $X^{\nabla\nabla\nabla} \subseteq \{(u, \tilde{u}), (\tilde{v}, v)\}^\nabla$, par transitivité $X^{\nabla\nabla\nabla} \subseteq (u, v)^\nabla$. Ainsi, en dérivant cette expression on obtient $(u, v)^{\nabla\nabla} \subseteq X^{\nabla\nabla\nabla\nabla}$. Finalement comme $(u, v) \in (u, v)^{\nabla\nabla}$ et $X^{\nabla\nabla\nabla\nabla} = X^{\nabla\nabla}$ on a $(u, v) \in X^{\nabla\nabla}$. □

On étudie dans un premier temps les cas particuliers contenant l'objet $\perp$ en commençant par le cas où cet élément apparaît dans les deux projections.

Proposition 5. *Si* $\perp \in \pi_1$ *et* $\perp \in \pi_2$ *alors* $X^\nabla = \emptyset$ *et* $X^{\nabla\nabla} = O_\bowtie$

Démonstration. Supposons $\perp \in \pi_1$ et $\perp \in \pi_2$ alors par définition de $\perp$ on a $X^\nabla \cap A_1 = \emptyset$ et $X^\nabla \cap A_2 = \emptyset$ et donc $X^\nabla = \emptyset$. Par définition de la dérivation on a $\emptyset^\nabla = O_\bowtie$ et donc $X^{\nabla\nabla} = O_\bowtie$. On montre la seconde assertion de manière symétrique. □

Dans le cas décrit par le lemme 5, il est immédiat de montrer que l'on peut construire $(X^{\nabla\nabla}, X^\nabla)$ depuis les éléments $\top$ des treillis de K_1 et K_2. On montre qu'il en est de même lorsque seule une des composantes π_1 ou π_2 contient $\perp$ en commençant par décrire X^∇ et $X^{\nabla\nabla}$ dans les lemmes 5 et 6.

Lemme 5. *Si* $\perp \in \pi_1$ *et* $\perp \notin \pi_2$, $X^\nabla = \pi_2'$. *Si* $\perp \in \pi_2$ *et* $\perp \notin \pi_1$, $X^\nabla = \pi_1'$.

Démonstration. Supposons $\perp \in \pi_1$ et $\perp \notin \pi_2$. On a $a \in X^\nabla$ si et seulement si $X^{\nabla\nabla} \subseteq a^\nabla$. Or, puisque $\perp \in \pi_1$ on a par construction $X^\nabla \cap A_1 = \emptyset$. Ainsi, on a $a \in A_2$. Donc, on a $a \in X^\nabla$ si et seulement si pour tout $(o_1, o_2) \in X^{\nabla\nabla}$ o_2 porte l'attribut a, c'est-à-dire $a \in \pi_2'$. Puisqu'on a $a \in X^\nabla$ si et seulement si $a \in \pi_2'$, on a $X^\nabla = \pi_2'$. On montre la seconde assertion de manière symétrique. □

Lemme 6. *Si* $\perp \in \pi_1$ *et* $\perp \notin \pi_2$, $X^{\nabla\nabla} = (O_1 \cup \{\perp\} \times \pi_2) \cap O_\bowtie$. *Si* $\perp \in \pi_2$ *et* $\perp \notin \pi_1$, $X^{\nabla\nabla} = (\pi_1 \times O_2 \cup \{\perp\}) \cap O_\bowtie$

Démonstration. Supposons $\perp \in \pi_1$ et $\perp \notin \pi_2$. Soit $v \in \pi_2$. Tout couple $(u, v) \in O_\bowtie$ vérifie $\pi_2' \subseteq (u, v)^\nabla$. Comme, par le lemme 5, on a $X^\nabla = \pi_2'$, on peut écrire $X^\nabla \subseteq (u, v)^\nabla$ et donc, par dérivation $(u, v)^{\nabla\nabla} \subseteq X^{\nabla\nabla}$. Enfin, pour tout $u \in O_1 \cup \{\perp\}$, on a $(u, v) \in X^{\nabla\nabla}$ donc $X^{\nabla\nabla} = (O_1 \cup \{\perp\} \times \pi_2) \cap O_\bowtie$. On montre la seconde assertion de manière symétrique. □

Les lemmes 5 et 6 nous permettent de déterminer que dans les cas où seule une des projections contient $\perp$ on peut écrire un concept formel de $K_\bowtie$ uniquement

avec l'autre projection. Étudions maintenant un concept formel basé sur cette projection déterminé par ARC

Lemme 7. *Si* $\perp \in \pi_1$ *et* $\perp \notin \pi_2$*, il existe un concept* $C_2 = (\pi_2, \pi_2')$ *sur* K_2*. Si* $\perp \in \pi_2$ *et* $\perp \notin \pi_1$*, il existe un concept* $C_1 = (\pi_1, \pi_1')$ *sur* K_1*.*

Démonstration. Supposons $\perp \in \pi_1$ et $\perp \notin \pi_2$. Comme $\pi_2 \subseteq O_2$, (π_2'', π_2') est un concept sur K_2. Il suffit donc de montrer que $\pi_2'' = \pi_2$, ou plus simplement $\pi_2'' \subseteq \pi_2$. Soit $o \in \pi_2''$. Par construction au moins un couple $(\bar{o}, o) \in O_\bowtie$ et $o' \subseteq (\bar{o}, o)^\nabla$. Or, on a $o \in \pi_2''$ donc par dérivation, $\pi_2' \subseteq o'$. De plus, par le lemme 5, on a $X^\nabla = \pi_2'$. Ainsi, $X^\nabla \subseteq (\bar{o}, o)^\nabla$ donc par dérivation, $(\bar{o}, o) \in X^{\nabla\nabla}$. Finalement, par définition des projections $o \in \pi_2$. On montre la seconde assertion de manière symétrique. □

La proposition suivante rassemble les lemmes précédents. Elle souligne que, dans le cas où une seule des deux projections contient $\perp$, tout concept de $K_\bowtie$ peut s'exprimer avec l'autre projection. De plus, il existe un concept généré par ARC, de même intension et dont l'extension correspond à une projection de l'extension du concept produit par AFC.

Proposition 6. *Soit* $C = (X^{\nabla\nabla}, X^\nabla)$*. Si* $\perp \in \pi_1$ *et* $\perp \notin \pi_2$*,* $C = ((O_1 \cup \{\perp\} \times \pi_2) \cap O_\bowtie, \pi_2')$ *et il existe un concept correspondant* $C_2 = (\pi_2, \pi_2')$ *sur* K_2*. Si* $\perp \in \pi_2$ *et* $\perp \notin \pi_1$*,* $C = ((\pi_1 \times O_2 \cup \{\perp\}) \cap O_\bowtie, \pi_1')$ *et il existe un concept correspondant* $C_1 = (\pi_1, \pi_1')$ *sur* K_1*.*

Démonstration. Découle des lemmes 5, 6 et 7. □

Il reste un cas spécifique, décrit par le lemme 8, pour compléter la description exhaustive d'un concept formel sur la table de jointure.

Lemme 8. *Pour tout* $X \subseteq O_1 \times O_2$ *on a* $X^\nabla = \pi_1' \cup \pi_2'$ *et* $X^{\nabla\nabla} = \{(o_1, o_2) \mid \pi_1' \subseteq o_1' \wedge \pi_2' \subseteq o_2'\}$

Démonstration. Montrons $X^\nabla = \pi'_1 \cup \pi'_2$ par double inclusion.
(i) $X^\nabla \subseteq \pi'_1 \cup \pi'_2$.

La modélisation de la RCF nous assure que $A_1 \cap A_2 = \emptyset$. Ainsi, un attribut $a \in X^\nabla$ est soit dans A_1, soit dans A_2. Si $a \in X^\nabla \cap A_1$, il doit être partagé par tous les éléments de π_1 ; et donc a est dans π'_1. De même, si $a \in X^\nabla \cap A_2$, $a \in \pi'_2$. On en déduit que $X^\nabla \subseteq \pi'_1 \cup \pi'_2$.

(ii) $\pi'_1 \cup \pi'_2 \subseteq X^\nabla$.

D'un autre côté, si un attribut a est dans π'_1, alors tout couple de $X^{\nabla\nabla}$ possède une première composante qui porte l'attribut a. Puisque cette propriété est vraie pour tout couple de $X^{\nabla\nabla}$ et que $X \subseteq X^{\nabla\nabla}$, alors tout couple de X porte l'attribut a. Donc, on a $a \in X^\nabla$. De la même manière, on montre que si $a \in \pi'_2$, alors $a \in X^\nabla$. Ainsi, on a $\pi'_1 \subseteq X^\nabla$ et $\pi'_2 \subseteq X^\nabla$. On peut donc affirmer que $\pi'_1 \cup \pi'_2 \subseteq X^\nabla$.

Finalement, par (i) et (ii) on a $X^\nabla = \pi'_1 \cup \pi'_2$. Comme $X^{\nabla\nabla}$ décrit exactement l'ensemble des couples (o_1, o_2) ayant les attributs de $\pi'_1 \cup \pi'_2$, par construction de la table de jointure on a $\pi'_1 \subseteq o'_1$ et $\pi'_2 \subseteq o'_2$ □

Les cas décrits par les lemmes 5 et 6 permettent de sélectionner de manière immédiate les concepts issus du processus d'ARC correspondant en terme d'extension à un concept de la table de jointure. La proposition 7 s'appuie sur le lemme 8 pour énoncer le résultat principal de cette sous-section, traitant des cas non dégénérés (sans élément $\perp$).

Proposition 7. *Soit $X \subseteq O_1 \times O_2$. Il existe par ARC sur K_1 un concept (X_1, Y_1) tel que $X_1 = \pi_1$ et $\pi'_1 \subseteq Y_1$ et il existe sur K_2 un concept (X_2, Y_2) tel que $X_2 = \pi_2$ et $\pi'_2 \subseteq Y_2$*

Démonstration. Comme $\pi_1' \subseteq A_1$ et $\pi_2' \subseteq A_2$, $C_1 = (\pi_1'', \pi_1')$ et $C_2 = (\pi_2'', \pi_2')$ sont des concepts formels sur leur contextes respectifs calculés à l'étape 0 de ARC.

Considérons les contextes K_1 et K_2 après graduation par l'opérateur $\exists$ sur les relations r et r^{-1}. On a alors l'attribut $\exists r : C_2$ dans K_1 et $\exists r^{-1} : C_1$ dans K_2. On définit les ensembles d'attributs $Y_1 = \pi_1' \cup \{\exists r : C_2\}$ et $Y_2 = \pi_2' \cup \{\exists r^{-1} : C_1\}$ ainsi que les concepts $C_3 = (Y_1', Y_1'')$ et $C_4 = (Y_2', Y_2'')$ (il est possible que $C_1 = C_3$ ou $C_2 = C_4$). On a $Y_1' = \pi_1'' \cap \{\exists r : C_2\}'$, montrons que $Y_1' = \pi_1$ par double inclusion.

Soit $o \in \pi_1$, on a $o \in \pi_1''$. De plus, par construction, tout couple de $(o, \bar{o}) \in X^{\nabla\nabla}$ vérifie $(o, \bar{o}) \in r$ avec $\bar{o} \in \pi_2$ et, par hypothèse, $\bar{o} \neq \perp$. Ainsi, puisque $\pi_2 \subseteq \pi_2''$, o porte l'attribut $\exists r : C_2$. Ainsi, on a $\pi_1 \subseteq Y_1'$.

Soit $o \in Y_1'$. On a $\pi_1' \cup \{\exists r : C_2\} \subseteq o'$. Comme $\{\exists r : C_2\} \subseteq o'$, il existe $\bar{o} \in \pi_2''$ tel que $(o, \bar{o}) \in r$ et donc $(o, \bar{o}) \in O_{\bowtie}$. De plus, comme $\bar{o} \in \pi_2''$, on a $\pi_2' \subseteq \bar{o}'$. Puisque $\pi_2' \subseteq \bar{o}'$, $\pi_1' \subseteq o'$ et que par le lemme 8 on a $X^{\nabla} = \pi_1' \cup \pi_2'$, on peut affirmer $X^{\nabla} \subseteq (o, \bar{o})^{\nabla}$. Finalement $(o, \bar{o})^{\nabla\nabla} \subseteq X^{\nabla\nabla}$ et par définition de π_1, on a $o \in \pi_1$. (De manière totalement analogue, on montre $\pi_2 = Y_2'$)

Finalement, on a montré l'existence de $C_3 = (Y_1', Y_1'')$ tel que $Y_1' = \pi_1$ et $\pi_1 \subseteq Y_1''$ ainsi que de $C_4 = (Y_2', Y_2'')$ tel que $Y_2' = \pi_2$ et $\pi_2 \subseteq Y_2''$ $\square$

En conclusion, les propositions 5, 6 et 7 montrent que pour tout concept $C = (X^{\nabla\nabla}, X^{\nabla})$ on trouve sur K_1 un concept (X_1, Y_1) tel que $X_1 = \pi_1$ et $\pi_1' \subseteq Y_1$ et il existe sur K_2 un concept (X_2, Y_2) tel que $X_2 = \pi_2$ et $\pi_2' \subseteq Y_2$. Il est simplement à noter que si $\perp \in \pi_1$ (respectivement π_2) on a $\pi_1 = \{x \mid \exists y, (x, y) \in O_{\bowtie}\}$ (respectivement $\pi_2 = \{x \mid \exists x, (x, y) \in O_{\bowtie}\}$). L'exemple 29 illustre ces propriétés.

Exemple 29. *Considérons la famille relationnelle ainsi que le contexte de semi-jointure définie à l'exemple 28.*

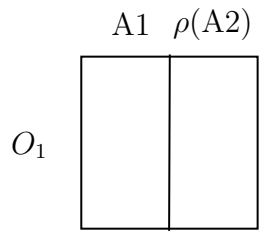


Figure 4.9: Modélisation après agrégation de la famille figure 4.7

Sur le contexte de jointure on trouve le concept $C = (\{(La, t3), (La, f5)\}, \{Adulte, Femme, Riche, pu\})$. *Ici,* $\pi_1 = \{La\}$ *et* $\pi_2 = \{t3, f5\}$. *On vérifie qu'il existe sur* K_1 *un concept* (π_1, π'_1), *à savoir le concept* $C_1 = (\{La\}, \{Adulte, Femme, Riche\})$, *et sur* K_2 *un concept* (π_2, π'_2), *le concept* $C_2 = (\{t3, f5\}, \{pu\})$.

Après une itération de l'ARC les intensions de ces concepts sont étendus. Ainsi, on a $C_1 = (\{La\}, \{Adulte, Femme, Riche, \exists pos : C_2\})$ *et* $C_2 = (\{t3, f5\}, \{pu, \exists pos^{-1} : C_1\})$.

4.4.2 Opération d'agrégation en cas mono-relationel

La sous-section 4.4.1 présente une analyse face à un cas de semi-jointure. Cette sous-section présente une comparaison avec un encodage plus proche de l'ARC. Prenons encore la même modélisation de RCF figure 4.7. On considère ici le cas concurrent où l'AFC est appliquée sur la modélisation de la figure 4.9. Cette modélisation consiste à étendre un contexte par des attributs d'un second contexte en relation avec le premier. Pour cela, conformément à la modélisation en RCF figure 4.7, on produit le contexte formel $K = (O, A, I)$ où $O = O_1$, $A = A_1 \cup \rho(A_2)$. $\rho(A_2)$ est l'ensemble des attributs de A_2 précédés d'un opérateur de graduation ρ quelconque, on notera un attribut de cet ensemble $\overline{\rho r : a}$, où $a \in A_2$, pour éviter la confusion avec les attributs relationnels produits par ARC. Pour un couple $(o_p, a_p) \in O \times A$, comme $A_1 \cap \rho(A_2) = \emptyset$ on a $a_p \in A_1$ ou (exclusif) $a_p \in \rho(A_2)$.

K_a	Senior	Adulte	Homme	Femme	Riche	Sportif	$\forall\exists pos : el$	$\forall\exists pos : pu$	$\forall\exists pos : cp$	$\forall\exists pos : nd$
Fa	×		×						×	×
La		×		×	×			×		
Sh		×		×					×	
Tr		×	×		×	×				

Tableau 4.5: Agrégation de la famille relationnelle de l'exemple 30

La relation d'incidence I sur ce contexte agrégé est la suivante :

— si $a_p \in A_1$ alors $(o_p, a_p) \in I$ si et seulement si (o_p, a_p) existe dans la relation d'incidence de T1

— si $a_p \in \rho(A_2)$ alors $(o_p, a_p) \in I$ si et seulement si $Contrainte(\rho, r, o_p, (a'_p, a''_p))$ est vrai (cf tableau 3.5).

L'exemple 30 illustre une telle modélisation avec l'opérateur $\forall\exists$ (le raisonnement avec un autre opérateur est similaire). Bien évidemment, pour une description des objets de O_2, il suffit d'inverser les rôles de K_1 et K_2 ainsi que de considérer la relation r^{-1} à la place de r.

Exemple 30. *Considérons à nouveau la famille relationnelle définie dans l'exemple 28. On agrège la famille en utilisant l'opérateur $\forall\exists$ donc pour un objet o de K_p si $\overline{\forall\exists pos : a} \in \rho(A_V)$ alors $(o, \overline{\forall\exists pos : a}) \in I$ si et seulement si o est lié à au moins un objet de K_v par la relation pos et que tous les objets liés à o portent l'attribut a.*

Le tableau 4.5 dépeint le contexte résultant de cette agrégation.

Comme à la sous-section 4.4.1, nous considérons les dérivations dans les contextes K_1 et K_2 toujours notées x', alors que la dérivation dans le contexte agrégé est notée x^{∇} (et la double dérivation $x^{\nabla\nabla}$).

Dans un premier temps, on définit un concept formel sur la table agrégée, puis on détermine le concept de même extension de la FRC. Enfin on présente la correspondance entre les intensions de ces deux concepts.

Pour définir un concept formel sur la table agrégée, on commence par identifier la composante de l'intension sur la partie $\rho(A_2)$ par la définition 14 et la proposition 8.

Définition 14. *Soit un ensemble d'objets quelconque $X \subseteq O$. On appelle* déviation relationnelle *de X l'ensemble des attributs sur $\rho(A_2)$ de X, c'est-à-dire $X^\nabla \cap \rho(A_2)$. On la note δ_X.*

Proposition 8. *Soit $X \subseteq O$. Sa* déviation relationnelle *est donnée par l'ensemble $\delta_X = \cap_{o \in X} \{\overline{\rho r : a} \mid Contrainte(\rho, r, o, (a', a''))\}$.*

Démonstration. Soit $o \in X$. Par construction, pour tout attribut $\overline{\rho r : a} \in \rho(A_2)$, on a $\overline{\rho r : a} \in o^\nabla$ si et seulement si $Contrainte(\rho, r, o, (a', a''))$. Ainsi $o^\nabla \cap \rho(A_2) = \{\overline{\rho r : a} \mid Contrainte(\rho, r, o, (a', a''))\}$. Comme $X' = \bigcap_{o \in X} o'$, on a $X' \cap \rho(A_2) = \bigcap_{o \in X} o^\nabla \cap \rho(A_2)$, donc $\delta_X = \cap_{o \in X} \{\overline{\rho r : a} \mid Contrainte(\rho, r, o, (a', a''))\}$ □

Un concept formel sur la table agrégée est alors donné par la proposition 9.

Proposition 9. *Soit un ensemble d'objets quelconque $X \subseteq O$. Le concept formel $C = (X^{\nabla\nabla}, X^\nabla)$ de la table agrégée vérifie $X^\nabla = X' \cup \delta_X$ et $X^{\nabla\nabla} = X'' \cap \{o \mid \delta_X \subseteq o^\nabla\}$.*

Démonstration. Par définition, on a $A = A_1 \cup \rho(A_2)$ et $A_1 \cap \rho(A_2) = \emptyset$. Ainsi on peut écrire $X^\nabla = (X^\nabla \cap A_1) \cup (X^\nabla \cap \rho(A_2))$, c'est-à-dire $X^\nabla = X' \cup \delta_X$.

En dérivant X^{∇}, on détermine que $X^{\nabla\nabla} = \{o \mid o \in O_1 \wedge X' \subseteq o^{\nabla} \wedge \delta_X \subseteq o^{\nabla}\}$. Or, comme $X' \subseteq A_1$, on a $X' \subseteq o^{\nabla}$ si et seulement si $X' \subseteq o^{\nabla} \cap A_1$, c'est-à-dire $X' \subseteq o'$. Finalement, $X^{\nabla\nabla} = X'' \cap \{o \mid \delta_X \subseteq o^{\nabla}\}$. □

Étudions le concept de même extension produit sur la FRC. Soit $X \subseteq O$. On note le i-ème élément de δ_X l' attribut $\overline{\rho r : a_{\delta,i}}$, où $a_{\delta,i} \in A_2$. Comme tous les concepts $C_i = (a'_{\delta,i}, a''_{\delta,i})$ existent sur le contexte K_2, après une étape de graduation via l'opérateur ρ sur la relation r, les attributs $\rho r : C_i$ sont ajoutés au contexte K_1. On pose $Y_\delta = X' \cup_{i \in 1..|\delta_X|} \rho r : C_i$. Les lemmes 9 et 10 montrent les liens entre Y_δ et $X^{\nabla\nabla}$, synthétisés dans la proposition 10.

Lemme 9. *On a $Y'_\delta \subseteq X^{\nabla\nabla}$.*

Démonstration. Soit $o \in Y'_\delta$. Premièrement o porte tous les attributs de X', c'est-à-dire $X' \subseteq o^{\nabla}$. De plus, pour chaque attribut $\overline{\rho r : a_{\delta,i}} \in \delta_X$ on peut trouver un concept $C_i = (a'_{\delta,i}, a''_{\delta,i})$ tel que o porte l'attribut $\rho r : C_i$, c'est-à-dire pour lequel on vérifie $Contrainte(\rho, r, o, C_i)$. Comme $a_{\delta,i}$ est dans l'intension de C_i, on peut vérifier que $\overline{\rho r : a_{\delta,i}} \in o^{\nabla}$. Puisque pour tout i, on a $\overline{\rho r : a_{\delta,i}} \in o^{\nabla}$, alors on a $\delta_X \subseteq o^{\nabla}$. Finalement, comme $\delta_X \subseteq o^{\nabla}$ et $X' \subseteq o^{\nabla}$, on a $X^{\nabla} \subseteq o^{\nabla}$. En dérivant cette inclusion on a $o^{\nabla\nabla} \subseteq X^{\nabla\nabla}$. Finalement, $Y'_\delta \subseteq X^{\nabla\nabla}$.

□

Lemme 10. *On a $X^{\nabla\nabla} \subseteq Y'_\delta$*

Démonstration. Soit $o \in X^{\nabla\nabla}$, alors premièrement o porte tous les attributs de X'. De plus, o porte chaque attribut $\overline{\rho r : a_{\delta,i}} \in \delta_X$, c'est-à-dire qu'on vérifie $Contrainte(\rho, r, o, (a'_{\delta,i}, a''_{\delta,i}))$. Puisque cette contrainte est vérifiée, o porte , après graduation de K_1, l'attribut $\rho r : (a'_{\delta,i}, a''_{\delta,i})$ pour tout $\overline{\rho r : a_{\delta,i}} \in \delta_X$. Ainsi, on a $Y_\delta \subseteq o'$ et donc $o'' \subseteq Y'_\delta$. Finalement, comme $o \in o''$ on conclut que $X^{\nabla\nabla} \subseteq Y'_\delta$. □

Proposition 10. *Les concepts* $C = (X^{\nabla\nabla}, X^{\nabla})$ *et* $C_\delta = (Y'_\delta, Y''_\delta)$ *sont de même extension.*

Démonstration. Découle des lemmes 9 et 10. □

La proposition 10 permet de déterminer que pour tout concept créé sur la table agrégée, un concept créé par ARC de même extension sera généré. On introduit l'affaiblissement relationnel à la définition 15, et illustré à l'exemple 31, pour exprimer la mise en correspondance entre les intensions de ces deux concepts. Cette dernière est énoncée dans la proposition 11.

Définition 15. *Soit un concept* C *produit par ARC et soit* Y_r *l'ensemble des attributs relationnel de l'intension de* C*. On appelle* affaiblissement relationnel *de* C*, noté* $\Omega(C)$*, l'ensemble*

$$\Omega(C) = \bigcup_{\rho r:(U,V)\in Y_r} \{\overline{\rho r : v} \mid v \in V\}$$

Exemple 31. *Considérons les contextes de l'exemple 20. On observe sur le contexte* K_V*, entre autres, les concepts* $C_1 = (\{t3\}, \{el, pu\})$*,* $C_2 = (\{f5\}, \{pu, cp\})$*,* $C_3 = (\{t3, zo\}, \{el\})$ *et* $C_4 = (\{zo, f5\}, \{cp\})$*. Après une graduation par l'opérateur existensiel, on observe sur* K_P *le concept* $C = (\{La\}, \{Adulte, Femme, Riche, \exists r : C_1, \exists r : C_2, \exists r : C_3, \exists r : C_4, \exists r : \top\})$*. Alors* $\Omega(C) = \{\exists r : el, \exists r : pu, \exists r : cp\}$*.*

Proposition 11. *On a* $\delta_X \subseteq \Omega(C_\delta)$

Démonstration. Notons Y_r l'ensemble des attributs relationnel de l'intension de C_δ. Soit $\overline{\rho r : a} \in \delta_X$. Alors par graduation et construction de C_δ, on a $\rho r : (a', a'') \in Y_r$ et comme $a \in a''$, on a $\delta_X \subseteq \Omega(C_\delta)$.

□

K_a	Senior	Adulte	Homme	Femme	Riche	Sportif	$\overline{\exists pos : el}$	$\overline{\exists pos : pu}$	$\overline{\exists pos : cp}$	$\overline{\exists pos : nd}$
Fa	×		×						×	×
La		×		×	×		×	×	×	
Sh		×		×				×	×	×
Tr		×	×		×	×				

Tableau 4.6: Agrégation de la famille relationnelle de l'exemple 32

Nous avons donc montré l'égalité des extensions de C et C_δ. Toutefois, les intensions ces deux concepts ne sont pas identiques. Comme le présente la proposition 11 et l'illustre l'exemple 32, l'intension du concept de la table agrégée est un sous-ensemble de l'affaiblissement du concept de même extension produit par l'ARC. En ce sens on peut considérer le concept produit par ARC plus informatif.

Exemple 32. *Supposons que l'on utilise la famille relationnelle définie à l'exemple 28 avec l'opérateur* $\exists$*. Le contexte agrégé est présenté au tableau 4.6.*

Puisque Farley (Fa) possède uniquement une Twingo (tw), par ARC on peut découvrir après une itération le concept $(\{Fa\}, \{Senior, Homme, \exists pos : (cp, nd)\})$ *alors que par AFC on trouve le concept* $(\{Fa\}, \{Senior, Homme, \exists pos : (cp), \exists pos : (nd))\})$*. L'attribut* $\exists pos : (cp, nd)$ *implique* $\exists pos : (cp)$ *et* $\exists pos : (nd)$*, mais* $\exists pos : (cp)$ *et* $\exists pos : (nd)$ *n'implique pas* $\exists pos : (cp, nd)$*.*

4.4.3 Généralisation à une famille multi-relationnelle

Supposons maintenant une RCF avec toujours deux contextes, mais plusieurs relations. Nous illustrons par un exemple. Prenons un contexte de patients, et un contexte de médicaments, ajoutons les relations $r_1 =$ "*prend*" qui spécifie si un patient prend un certain médicament et $r_2 =$ "*allergique*" qui spécifie lorsqu'un

médicament est contrindiqué pour un patient.

Considérons maintenant la première modélisation proposée, la jointure, et montrons que la généralisation dans un cadre à multiple relations n'est pas automatique. Soit un patient p_1 qui prend le médicament m_1 et est allergique au médicament m_2. Supposons que m_1 et m_2 partagent l'attribut a. Le concept sur le contexte agrégé (a', a'') existe, et a' contient autant le couple (p_1, m_1) que (p_1, m_2). L'information sur le fait que p_1 soit lié à un médicament ayant a est conservé mais, on ne peut plus savoir s'il y a une réaction allergique ou si le médicament prescrit. Cette modélisation simple ne peut donc pas être adaptée à un cas à multiple relations.

Supposons maintenant que l'on décide de rajouter un ensemble de colonnes pour chaque relation existante comme dans la figure 4.10. Ainsi on peut spécifier la relation qui lie un couple d'objet de $O_{\bowtie}$. Cette modélisation ne change pas la propriété précitée. En effet, si l'on se replace dans le cadre du paragraphe précédent, le concept (a', a'') n'a pas de relation dans son intension et le problème persistera.

La seule solution semble être la duplication des attributs, annotés par la relation reliant les objets des deux tables. Une représentation se trouve à la figure 4.11. Dans un cas présentant n relations, $O_{\bowtie}$ décrit un ensemble de $n+1-uplets$ de la forme $(o_0, o_1, ..., o_n)$ où $o_0 \in O_1 \cup \{\perp\}$ et pour $i \in 1..n$, $o_i \in O_2 \cup \{\perp\}$. La valeur de la relation n'est alors pas perdue.

Dans ce cas, la démonstration présentée en 4.4.1 est toujours valide, sous réserve de construire tous les concepts dans ARC correspondant aux $n+1$ projections sur les différentes composantes des tuples de $O_{\bowtie}$ et de créer tous les attributs relationnels vers les fermetures de chacunes de ses projections. Ainsi, en environnement à multiples relations, la propriété soulevant que «tout concept créé l'AFC sur un contexte créé par jointure d'une RCF peut être retrouvée par ARC» est vraie.

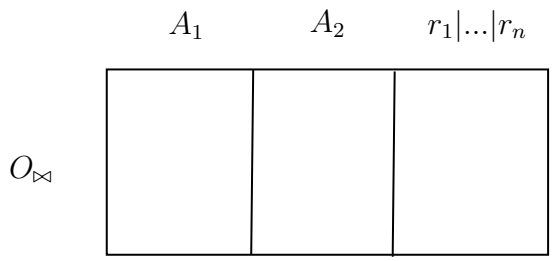


Figure 4.10: Modélisation jointure mutiples relations

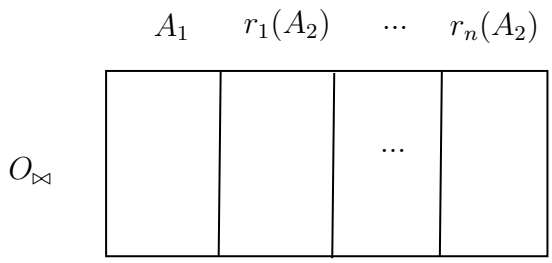


Figure 4.11: Modélisation jointure mutiples relations par duplication

On peut aussi montrer, par induction, que si le nombre de contextes est supérieur à 2, la propriété est toujours vraie : supposons qu'elle soit vraie pour $n-1$ contexte, il suffit alors de considérer le contexte join comme la résultante d'un contexte (qui est lui même la jointure d'une famille de $n-1$ contextes) et d'un nouveau $n-$ième contexte et réitérer le raisonnement.

Il en va de même pour la modélisation par agrégation.

4.5 Conclusion

Dans ce chapitre, nous avons formellement démontré l'intérêt de l'ARC par rapport à l'application de l'AFC sur un jeu de données relationnel, après agrégation des contextes. Nous avons étudié les limites du formalisme du chapitre 3, notamment la définition de concept formel dans un cadre de contextes étendus de manière itérative. Nous avons démontré qu'avec un formalisme plus exhaustif, les règles d'associations pouvaient être extraites et définies sans attribut à résoudre

récursivement. Nous avons aussi montré certaines optimisations, inhérentes aux opérateurs de propositionnalisation.

La partie II montre l'intérêt de ces optimisations et des règles d'association au travers de trois cas d'application. Le chapitre 5 utilise les règles d'association pour analyser les arrêts de production dans l'industrie métallurgique. Le chapitre 6 présente un cas d'application en psycholinguistique dans lequel les règles d'association permettent d'analyser la structure de dictionnaires. Finalement, le chapitre 7 est une étude préliminaire montrant les possibilités de l'ARC dans le domaines de l'ingénierie de connaissance.

Deuxième partie

Applications

CHAPITRE V

AIDE À LA PRISE DE DÉCISION INDUSTRIELLE

Ce chapitre se consacre à la présentation d'un cas d'étude issu du milieu de production industrielle. Nous étudions ici une chaîne de pressage visant à produire des poignées et des cadres en aluminium pour des fenêtres et des portes.

5.1 Introduction

L'une des pierres angulaires de la prise de décision managériale en industrie est le coût des opérations, notamment ceux associés aux problèmes de production et à leur résolution. Pour déterminer les différentes solutions possibles, les experts du domaine doivent, d'une part, caractériser les différents problèmes et, d'autre part, comprendre leurs origines. Pour cela, il s'appuient sur un modèle causal, lui-même raffiné au cours du temps selon les différentes observations faites sur la chaîne de production. Les différentes observations reflètent les co-occurrences entre divers éléments sur la chaîne de production. Les alertes signalées par les différents capteurs des machines de production et l'état des pièces manufacturées sont les principaux indicateurs étudiés. Les règles d'association présentant les co-occurrences d'un jeu de données, nous explorons donc comment leur extraction peut aider à la détection et à la compréhension des anomalies dans un tel contexte.

5.2 Cadre d'étude

Dans l'entreprise à la source des données que l'on utilise ici [1], les machines de production sont imposantes. De fait, tout arrêt, qu'il soit lié à une panne ou à un entretien, est coûteux. Pour minimiser ces arrêts, l'entreprise a adopté une politique d'arrêt uniquement en cas de production de pièces non conformes aux critères de qualité (par exemple, l'épaisseur de la pièce). Lorsqu'un tel cas se produit, un opérateur stoppe manuellement la machine, et après un diagnostic succinct, réalise les tâches d'entretien (ou de réparation) adéquates, puis redémarre la production. Cette procédure minimise certes les arrêts, et maximise donc le temps d'activité de la machine ; néanmoins, la production d'une pièce en aluminium défectueuse demande une refonte complète et une réédition de la pièce pour maintenir une qualité de produit conforme. Ce processus est coûteux en terme de temps, ce qui, en pratique, se traduit par une perte financière. La figure 5.1 [2], adaptée de (Atkinson, 1999) illustre le fait que la qualité, le coût et la rapidité de production s'opposent : prioriser l'un de ces objectifs se fait au détriment des deux autres. On vise à établir un modèle qui participe à l'optimisation du point d'équilibre de l'entreprise en réduisant les coûts de production sans affecter la qualité et le temps de production. Pour cela, on cherche les éléments déclencheurs des problèmes pour créer, en conséquence, un processus qui minimise les arrêts forcés, et qui évite la mise au rebut d'une pièce.

---

1. Le nom de l'entreprise doit rester anonyme dans cette thèse.

2. Les figures de ce chapitre sont extraites du poster présenté à la conférence K-CAP '19.

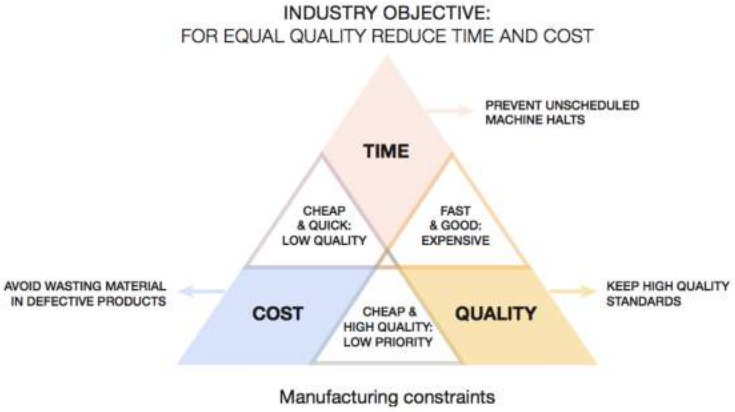


Figure 5.1: Triangle des contraintes de production.

5.3 Modélisation du problème de maintenance

Une analyse en profondeur des données par des techniques de fouille permet de découvrir des régularités dans un grand jeu de données. Celles-ci seront exprimées au travers de règles d'association, liant les variations qui peuvent être trouvées entre les différents produits, les états de la machine et les problèmes de production.

Principalement, notre jeu de données est composé de deux types d'objets : un type présente les problèmes rencontrés, un autre présente les valeurs des capteurs de notre ligne de production. Une relation binaire de co-occurrence lie les états de la machine aux problèmes. Son inverse est également exploité. Ces relations sont de type plusieurs-à-plusieurs. En effet, un état (ensemble des valeurs captées) peut arriver conjointement avec plusieurs problèmes et un problème peut apparaitre pour différents états.

Le fait que la relation soit de type plusieurs-à-plusieurs a fait, de manière très naturelle, pencher la balance en faveur d'une méthode de fouille de données multi-relationnelles (FDMR). Parmi les méthodes de FDMR, le choix s'est porté sur l'ARC puisqu'elle permet aussi l'extraction de règles d'association relationelles de manière non supervisée, qui, telles que définies au chapitre 3, ne sont pas

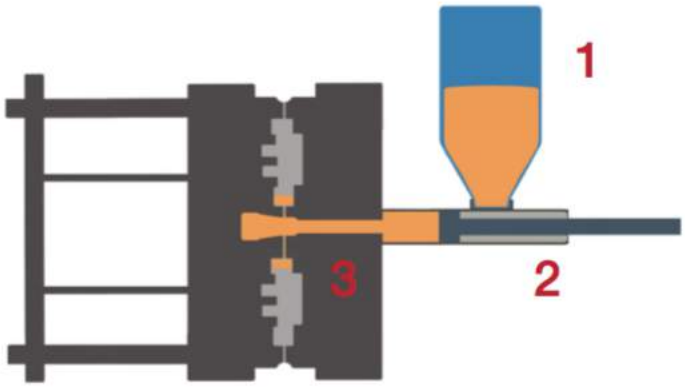


Figure 5.2: Schéma de la machine, avec annotation des étapes de production

contraintes sur le nombre d'éléments en prémisse et conclusion, contrairement aux autres méthodes présentées au chapitre 1.

5.3.1 Données

Les données utilisées ont été recueillies pendant un mois complet de production. Elles traduisent les valeurs affichées par les capteurs lors de la manufacture. La figure 5.2 illustre le processus de transformation de l'aluminium en pièces manufacturées. Il se fait en trois étapes à partir du moment où l'aluminium est placé dans le compartiment de départ :

1. La cuve chauffe et le métal y est fondu.
2. Un piston injecte le contenu de la cuve dans le moule.
3. Une fois le liquide dans le moule, le piston continue à presser jusqu'à refroidissement et solidification.

Après plusieurs modélisations, le jeu de données final que nous considérons couvre deux types d'objets :

— les données machines liées à la production d'une pièce, qu'on appellera les *données pièces*. Sur le mois complet, environ 58 000 pièces ont été produites.

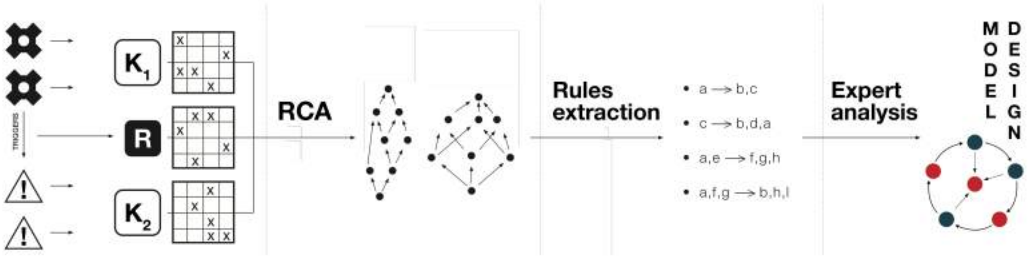


Figure 5.3: Processus d'analyse des données

Pour chacune de ces pièces, 25 caractéristiques ont été enregistrées. Elles regroupent : la date et la période horaire de la production ; le type de moule utilisé pour concevoir la pièce ; la conformité, l'épaisseur et le volume finaux de celle-ci ; la vitesse, la pression et la position du piston à la fin de chacune des phases décrites ci-dessus.

— 19 problèmes de production sont aussi enregistrés. Ils sont caractérisés par leur origine (piston, usure, calibrage...), leur répercussion financière ou répercussion temporelle.

Ces deux types d'objets sont en relation. Celle-ci spécifie, pour chaque pièce éditée, tous les problèmes (s'il en existe) qui ont été détectés lors de sa production.

Les jeux de données sont ensuite binarisés en contextes formels : les variables catégoriques sont transformées en autant de colonnes que de valeurs possibles, alors que les variables numériques sont discrétisées sur cinq intervalles égaux.

Ainsi, nous pouvons résumer le processus de traitement au travers de la figure 5.3 : Après récupération des données, celles-ci sont transformées en famille relationnelle de contexte ; l'ARC, dans sa version «sans cycle», est alors appliquée pour extraire des règles d'association ; ces dernières sont ensuite transmises à des experts qui les utilisent pour raffiner ou construire un modèle qui servira à la prise de décision.

5.4 Expérience 1 : Restriction aux pièces problématiques

5.4.1 Conception de la famille relationnelle

Dans une première expérimentation (Wajnberg *et al.*, 2019a), le découpage des variables sélectionnées par les experts Semeraro et al. a résulté en 134 attributs pour la table des pièces, et 26 pour la table des problèmes (Semeraro *et al.*, 2019).

Parmi les attributs caractérisant les pièces on compte $sm_{min} < LimLow$ signifiant que l'endroit où la pièce est la moins épaisse a une épaisseur inférieure au seuil d'épaisseur critique bas $LimLow$. $LimUp$ est le seuil critique haut. Une pièce est conforme si toutes ses valeurs sont comprises entre les seuils critiques haut et bas. L'attribut sm_{ko} signale sans précision que l'épaisseur d'une pièce est défectueuse. L'attribut $recast$ signale qu'une pièce devra être refondue. De plus, des attributs de la forme $c_{phase} = position$ (par exemple $c_1 = 83$) renseigne sur la position, en millimètres, du piston par rapport à la machine à la fin d'une des phases de production. La première section du tableau 5.1 répertorie ces différents attributs

Parmi les attributs sur les problèmes $\mu stop$ précise qu'un court arrêt est déclenché par le problème, $t_{<Y}$ détaille que le temps d'un arrêt est inférieur à Y minutes, $cost_{med}$ renseigne sur l'impact financier d'un problème et le qualifie de coût moyen, qlt indique qu'un problème est de type «problème de qualité», alors que qlt_{sm} précise qu'il est relatif à l'épaisseur entre les faces du moule (impactant les dimensions d'une pièce produite). L'attribut $predict$ signale qu'un problème était prévisible et aurait pu être pris en charge en amont par un opérateur. Enfin $sm = 14$ signale que l'un seuil d'épaisseur critique, fixé à 14mm, est atteint par une pièce produite. La seconde section du tableau 5.1 récapitule ces attributs.

La relation de co-occurrence avec les problèmes, notée g, a été combinée à l'opérateur $\exists$. Une interprétation générique d'un attribut relationnel ainsi générée serait

attribut	sémantique
$sm_{min} < LimLow$	pièce trop fine
$sm_{max} > LimUp$	pièce trop épaisse
sm_{ko}	pièce invalide
$recast$	pièce à refondre
$c_1 = X$	piston à Xmm de la machine lorsque l'aluminium passe de l'injecteur à l'entrée du moule
$c_2 = X$	piston à Xmm de la machine lorsque l'aluminium vient d'être complètement inséré dans le moule
$c_3 = X$	piston à Xmm de la machine à la fin de la phase de compression de l'aluminium dans le moule
$\mu stop$	problème déclenchant un arrêt court
$t_{<Y}$	problème résolu en moins de Y min avec $Y \in \{5, 10, 15, 30, 60\}$
$cost_Y$	impact financier du problème sur la production $Y \in \{low, med, high\}$
qlt	problème de qualité
qlt_{sm}	problème d'espacement dans le moule
$predict$	problème prévisible
$sm = 14$	production d'une pièce à la limite critique d'épaisseur

Tableau 5.1: Attributs de la famille relationnelle de la première expérimentation

la suivante : si un concept des *pièces* C porte l'attribut $\exists g : (\bar{C})$ dans son intension, alors toutes les pièces dans l'extension de C co-occurrent avec au moins un problème qui a tous les attributs de l'intension de $\bar{C}$. La relation inverse g^{-1} est, quant à elle, combinée à l'opérateur $\forall\exists$. Si un concept des *problèmes* C présente $\forall\exists g^{-1} : (\bar{C})$ dans son intension, alors les problèmes de C sont uniquement reliés aux pièces usinées ayant au moins tous les attributs de l'intension de $\bar{C}$. Notre choix s'est porté sur l'opérateur $\forall\exists$ afin de produire un résumé des caractéristiques partagées par toutes les pièces, plutôt que d'avoir une multitude de cas avec $\exists$.

K_1	grand	défectueux
o_1	×	
o_2	×	
o_3	×	
o_4		×

g	p_1	p_2	p_3	p_4	p_5	p_6
o_1	×			×		
o_2		×				×
o_3	×					
o_4			×		×	

K_2	$cout_{haut}$	$cout_{bas}$	$temps_{haut}$
p_1	×		
p_2	×		
p_3		×	
p_4		×	×
p_5		×	
p_6			×

Tableau 5.2: Famille relationnelle de l'exemple 33

Exemple 33. *On considère la famille relationnelle présentée au tableau 5.2. Le contexte K_1 présente quatre objets manufacturés o_1 à o_4 avec les attributs stipulant si une pièce est grande et défectueuse. Le contexte K_2 répertorie les problèmes p_1 à p_6 et les attributs de coût haut et bas induits par un problème, ainsi que l'attribut spécifiant que le temps de résolution est long.*

À l'itération 0 de l'ARC, on a sur K_1 le concept $C_O = (\{o_1, o_2, o_3\}, \{grand\})$ et sur K_2 le concept $C_P = (\{p_1, p_2\}, \{cout_{haut}\})$. De plus, on a les relations suivantes : $g(o_1) = \{p_1, p_4\}$ $g(o_2) = \{p_2, p_6\}$, $g(o_3) = \{p_1\}$,$g^{-1}(p_1) = \{o_1, o_3\}$ et $g^{-1}(p_2) = \{o_2\}$. Ainsi, comme pour toute pièce de C_O il existe au moins un problème co-occurrant qui soit dans C_P, C_O récupère , après une itération dans l'ARC, l'attribut $\exists g : C_P$. Cet attribut, par le processus d'explicitation présenté au chapitre 3, est résolu en $\exists g : (cout_{haut})$. Ainsi, la règle d'association de confiance 100% : $grand \rightarrow \exists g : (cout_{haut})$ est produite et signifie que tout objet qualifié de grand est co-occurrant avec au moins un problème ayant l'attribut $cout_{haut}$.

À l'inverse, tout problème de C_P compte parmi ses objets générateurs uniquement des objets présents dans C_O. Ainsi, après une itération, C_P contient l'attribut

$\forall\exists g^{-1}$: C_O *qui est résolu en* $\forall\exists g^{-1}$: $(grand)$. *Ainsi, il est possible d'extraire la règle d'association de confiance* 100% *:* $cout_{haut} \rightarrow \forall\exists g^{-1}$: $(grand)$ *qui signifie qu'un problème provoquant un coût financier élevé est systématiquement co-occurrant à la production d'objets de grande taille.*

5.4.2 Protocole

Dans notre première expérience, plusieurs filtres ont été appliqués pour limiter le nombre de règles produites par la fouille par ARC. Tout d'abord, nous avons restreint le processus d'ARC aux objets qui sont en relation avec au moins un problème. Le filtre appliqué a conservé environ 6000 objets. En premier lieu, nous nous sommes limité aux règles d'implication (règle d'association de confiance 100%) pour observer le comportement général du processus de production. De plus, comme nous nous intéressons aux liens entre attributs sur les pièces et les problèmes, seules les règles comportant au moins un attribut relationnel ont été générées. Finalement, la confiance étant fixée, les règles produites sont évaluées uniquement en considérant le support. L'étude des règles est donc effectuée selon un ordre décroissant sur le support. Ainsi, les règles de petit support (inférieur à 20%) ne sont pas consultées en priorité et ont donc été omises dans cette première expérience pour un gain de temps de calcul. Le traitement a été effectué sur un Macbook Pro 2012, avec 8Go de RAM et un processeur 2.3GHz Intel Core i7.

5.4.3 Résultats

De cette étude, nous avons extrait deux formes de règles : l'une sur l'épaisseur des pièces, l'autre sur les déplacements du piston.

5.4.3.1 Épaisseur invalide

Dans un premier temps, nous avons trouvé, l'ensemble de règles suivantes :

$$\begin{cases} (1)\, sm_{min} < LimLow \rightarrow sm_{ko} \\ (2)\, sm_{ko} \rightarrow sm_{min} < LimLow \\ (3)\, sm_{ko} \rightarrow \exists g : (\mu stop, qlt_{sm}, t_{<5min}, cost_{med}) \\ (4)\, sm_{ko} \rightarrow recast \end{cases}$$

Cet ensemble de règles peut être interprété comme suit :

Si un produit a une épaisseur qui ne répond pas aux exigences de qualité sm_{ko}, alors un problème sera signalé. Celui-ci viendra toujours du fait que l'épaisseur est en dessous du seuil bas (et pas au dessus du seuil haut) (1 et 2). Et, dans un tel cas, la pièce sera (4) refondue. De plus, ce problème déclenchera (3) un arrêt court (micro-stop) et lancera un signal d'alerte sur la qualité type épaisseur (sm-quality). Ce type de problème prend moins de 5 min ($t_{<5min}$) à résoudre et entraîne un impact financier moyen ($cost_{med}$).

5.4.3.2 Déplacement irrégulier du piston

Le déplacement du piston dans la pompe est reflété par trois variables indiquant sa position, c_1, c_2 et c_3.

Parmi les règles extraites par ARC, nous avons détecté un ensemble récurrent de règles liées à un même concept, où x, y et y sont des variables d'entiers, de la forme suivante :

$$\begin{cases} c_1 = x, c_3 = y \Rightarrow c_2 = z \\ c_2 = z, c_3 = y \Rightarrow c_1 = x \\ c_1 = x, c_2 = z \Rightarrow c_3 = y \\ c_1 = x, c_2 = z \Rightarrow \exists g\, pb : (sm = 14, predict, qlt) \end{cases}$$

Chaque fois que cet ensemble de règles apparaît, seule la combinaison de x, y et y varie. Toutefois, une étude supplémentaire montrait que bien que les valeurs variaient, la somme $x + y + z$ restait constante et x était en dessous d'un seuil critique.

5.4.3.3 Impact

Ces règles, après avoir été fournies à des experts du domaine, ont reflété la conclusion suivante. En constatant que l'épaisseur était toujours liée à un problème de seuil bas, les experts ont pu déterminer la cause du problème : des résidus d'aluminium se logeaient dans le moule de la machine de manière cumulative, un peu plus à chaque itération. À mesure que les résidus grandissaient, l'espace disponible à l'édition des pièces se réduisait, se traduisant par un déplacement anormal du piston. Cela résultait en des arrêts coûteux et des refontes de pièces (puisque celles-ci échouaient le contrôle-qualité de l'épaisseur). De ce constat, une solution a alors pu émerger, moins coûteuse et moins chronophage : calibrer les capteurs pour répondre automatiquement aux déviations de déplacement du piston. Cela permet d'envoyer des alertes aux opérateurs pour nettoyer la machine, juste avant de produire une pièce défectueuse. Ainsi, le nombre d'arrêts sera égal à celui actuellement en place, mais aucune pièce ne sera à refondre.

5.5 Expérience 2 : Sans contrainte

5.5.1 Conception de la famille relationnelle

La première expérience ne concernait que les objets ayant au moins un lien vers un problème. Ainsi, les descriptions/règles que nous avons pu établir décrivent entièrement les conditions qui existent chez de tels objets. Toutefois, l'absence des pièces «saines» ne permet pas de statuer sur le fait qu'une règle s'applique uniquement sur des produits défectueux ou si la règle s'applique dans tous les cas. C'est pourquoi, dans cette seconde expérience (Wajnberg *et al.*, 2019b), nous avons retiré la contrainte de n'étudier que les objets produits ayant au moins un problème. Les 58 000 pièces ont donc été prises en compte.

De plus, les attributs qui étaient affectés à des valeurs exactes, comme les positions de piston, ont été regroupés par intervalles. Les attributs sont identifiés par : $a_{i;j;k}$ où a est la caractéristique considérée, i est l'étape de la manufacture (voir section 5.3.1), j est l'intervalle (1 étant celui ayant les valeurs les plus basses, 5 les plus hautes). Une description complémentaire pour l'intervalle j est donnée par la valeur k. Elle indique si l'intervalle j va introduire un problème (ko) ou non (ok) ; *check* ne donne pas d'information systématique et demande une vérification au cas par cas. Parmi les attributs a possibles, on compte sm l'épaisseur de la pièce, c la position du piston, v sa vitesse, t le temps du piston dans une phase et p la pression. Le tableau 5.3 récapitule ce découpage. Les autres attributs comptent *recast* et *conform* qui indiquent si les pièces éditées pendant le problème vont devoir être refondues ou non. Différent codes de moules tels que $10A026$ sont aussi présents.

Les problèmes ont, en plus des caractéristiques de la première expérience, tous un attribut d'identification unique.

variable	valeur	sémantique
a	sm	épaisseur de la pièce
	c	position du piston en fin de phase
	v	vitesse du piston en fin de phase
	t	temps de la phase
	p	pression du piston en fin de phase
i	1	phase de fonte et d'injection dans la pompe
	2	phase de remplissage du moule
	3	phase de compression
j	1 à 5	intervalle de la valeur de a_i
k	ok	valeur normale
	ko	valeur anormale
	$check$	valeur limite

Tableau 5.3: Détail des attributs $a_{i;j;k}$

5.5.2 Résultats

Parmi les 133821 règles produites par notre outil, nous avons sélectionné celles qui semblaient avoir le plus de valeur auprès des experts industriels. La table 5.4 montre certaines des règles sélectionnées, réparties selon plusieurs groupes, que nous détaillons et interprétons dans les prochaines sous-sections.

5.5.2.1 Problèmes multiples

L'outil a extrait 3950 règles reflétant le fait qu'une même pièce usinée puisse être reliée à différents problèmes. Cette situation est assez rare, bien que non ponctuelle, ce qui rend l'information pertinente pour les experts. Les règles leur permettent d'observer, d'une part, les co-occurrences d'attributs qui déclenchent ces problèmes, mais surtout, de revoir une partie de la modélisation. En effet, de ces règles, les experts ont pu établir que certains problèmes, supposément séparés, étaient, en fait, le même problème.

#	Prémisse	Conclusion
1	$sm_{3;1;ko}, sm_{min}(< LimLow)$	$recast, \exists g\, pb : (sm \leq 14)$, $\exists g\, pb : (\mu stop_{smLimLow}, predict)$
2.1	$c_{2;1;ko}, t_{2;1;ko}, recast$	$c_{1;5;ko}, v_{1;5;ko}$, $\exists g\, pb : (\forall \exists g^{-1}\, pc : (recast))$
2.2	$c_{2;1;ko}, t_{2;1;ko}$, $\exists g\, pb : ($ $\forall \exists g^{-1}\, pc : (recast))$	$c_{1;5;ko}, v_{1;5;ko}, recast$
2.3	$v_{1;5;ko}, t_{2;1;ko}, recast$	$c_{1;5;ko}, c_{2;1;ko}$, $\exists g\, pb : (\forall \exists g^{-1}\, pc : (recast))$
2.4	$v_{1;5;ko}, t_{2;1;ko}$, $\exists g\, pb : (\forall \exists g^{-1}\, pc : (recast))$	$c_{1;5;ko}, c_{2;1;ko}, recast$
3	$\exists g\, pb : (rul_{component}, t_{\geq 1h}, phm)$	$conform$
4	$v_{2;4;ko}, p_{3;2;ok}$, $\exists g\, pb : (\forall \exists g^{-1}\, pc : (10A026, c_{1;5;ko}$, $c_{2;1;ko}, v_{1;5;ko}, t_{2;1;ko}$, $c_{c;2;ok}, pm_1, sm_{3;2;check}))$	$10A026, c_{1;5;ko}, c_{2;1;ko}, v_{1;5;ko}$, $t_{2;1;ko}, c_{c;2;ok}, pm_1$, $sm_{3;2;check}, conform$

Tableau 5.4: Extrait des règles produites par ARC

Regardons de plus près la règle #1 de la table 5.4. Son support est de 7%. Elle nous apprend que toute pièce usinée, qui atteint la borne basse ($sm_{min}(< LimLow)$) du plus petit des intervalles qualifiant l'épaisseur ($sm_{3;1;ko}$), sera une pièce défectueuse qui devra être refondue. Sans surprise, une alerte statuant que l'épaisseur est sous le seuil critique de 14mm sera lancée. Mais en plus, on peut observer que, dans ce cas, un micro-arrêt (arrêt de durée inférieure à 5 min) va être enregistré. Toutefois, de cette relation de co-occurrence, les experts ont pu s'interroger sur le fait qu'un produit, malgré un moule complètement rempli, ne fasse pas l'épaisseur minimale requise. Ils ont donc pu rapidement confirmer qu'il s'agissait du fait que l'aluminium restait collé au moule, et ne se séparait pas avec la pièce. Ce dépôt résiduel prend de la place dans le moule et s'accroît régulièrement, en cas d'attache de l'aluminium au moule. Le volume occupé est autant de place indisponible pour la pièce. Le résidu empêche ainsi la pièce d'atteindre le volume attendu.

5.5.2.2 Prévision des refontes

Un autre aspect intéressant pour les experts est d'étudier les situations qui engendrent des refontes. Dans un premier temps, nous avons fait une approche directe et naïve visant à sélectionner toutes les règles qui comprennent l'attribut de refonte (*recast*), qu'il soit attribut direct ou inclus dans la composition d'un attribut relationnel. Ainsi, ont été trouvées 42060 règles, dans lesquelles nous avons trouvé des sous-groupes partageant une origine commune. Ces sous-groupes sont caractérisés par le fait d'être des règles issues du même concept. Le lot #2 dans la table 5.4 est un exemple d'un tel groupe ; son support étant de 14%, il permet d'affirmer que le phénomène décrit est récurrent et mérite attention.

Bien que l'interprétation en profondeur du phénomène industriel ne soit pas dans nos objectifs, nous pouvons résumer le contenu de ces règles de la manière suivante : tout d'abord, il faut garder à l'esprit que, séparément, ces règles ne sont pas très explicites, ni particulièrement pertinentes. Toutefois, conjointement, elles exhibent les combinaisons d'attributs produisant une refonte.

Une première remarque peut être faite : une très basse température ($t_{2;1;ko}$) est toujours observée. Ensuite, on peut voir que pour être co-occurante d'une erreur, il faut que cette température soit combinée : soit à une grande vitesse du piston à la première étape ($v_{1;5;ko}$) ; soit, alternativement, que le piston se trouve trop proche de la machine à la fin de la phase 2 ($c_{2;1;ko}$). Les explications des experts viennent alors compléter notre description. Tout d'abord, il y a une équivalence entre cette trop grande vitesse et cette position avancée. En effet, si le piston va plus vite dans le même laps de temps, il sera plus rapproché de la machine. Dans la même logique, une des conclusions invariables de ces règles est que le piston est aussi trop avancé à la fin de la phase 1 ($c_{1;5;ko}$).

En prenant en compte ces combinaisons d'attributs et leurs entrecroisements dans les règles, les experts ont pu, du lien de co-occurrence, entrevoir la relation de causalité, ou, tout au moins, émettre une hypothèse qu'ils pourront tester : lors d'un redémarrage de la machine ou de tout autre évènement amenant des variations de températures notables sur la machine, il est possible que même si la température de la cuve est assez haute pour que l'aluminium fonde, la température du piston, elle, soit trop basse. Or, un objet chauffé se dilate, et refroidi se contracte. L'hypothèse émise est donc que le piston n'est pas assez dilaté, justifiant une friction trop peu intense, ce qui expliquerait l'augmentation de sa vitesse. De plus, si le piston n'est pas assez dilaté, il est possible que l'aluminium passe sur le côté et ne soit pas pressé, ou que la vitesse trop grande du piston empêche une évacuation d'air, ce qui serait néfaste pour la production de la pièce. Ainsi, une pièce défectueuse serait produite et devra être refondue.

5.5.2.3 Pièces conformes

Sachant que toutes les règles impliquant un problème n'apportent pas toujours l'attribut de refonte, les experts étaient particulièrement curieux à l'idée de découvrir les conditions sur les pièces qui co-occurrent avec un problème non critique ; c'est-à-dire un problème n'empêchant pas les pièces de passer les tests de conformité avec succès. Trouver de telles combinaisons est primordial pour les experts, puisqu'elles seraient des indicateurs préalables à l'arrivée de problèmes critiques. Dans la plupart des cas, les pièces, bien que conformes, ont certaines de leurs caractéristiques présentant des valeurs proches des limites de conformité. Déterminer ces combinaisons caractéristiques serait une avancée significative pour les industriels. Nous avons donc recherché les règles comprenant l'attribut de conformité $conform$. Nous avons trouvé 53008 règles.

Par exemple, la règle #3, dans la table 5.4, indique que si une pièce est liée à un problème mécanique ($rul_{component}$) qui met la machine à l'arrêt pendant plus d'une heure ($t_{\geq 1h}$) et que ce problème apparaît pendant une phase de maintenance, alors le produit resterait conforme. Cette information est cohérente puisque la machine est arrêtée manuellement pour la maintenance, traduisant une erreur minime (sinon l'arrêt aurait été déclenché automatiquement).

5.5.2.4 Aide à la modélisation

Enfin, en étudiant les pièces conformes, nous avons découvert un groupe de règles qui, plutôt que de donner de l'information sur les problèmes ou les pièces, ont mis en exergue des omissions dans les descriptions des problèmes. En effet, on peut voir dans la règle #4 de la table 5.4 (dont le support est juste en dessous de 1%), que certains attributs sont accompagnés d'un attribut relationnel. Cet attribut relationnel, cependant, renvoie à un concept supérieur (plus haut dans le treillis). La figure 5.4 illustre la situation. Le fait qu'aucun attribut non relationnel ne soit présent dans le concept des problèmes montre qu'il existe au moins deux problèmes concernés. Les experts ont été surpris de voir que ces problèmes, bien que liés à des pièces aux profils très similaires (huit attributs communs) ne partagent aucun attribut statique. Ceci est potentiellement une indication que des caractéristiques, communes à ces problèmes, manquent dans leur description.

5.6 Conclusion

Dans ce chapitre, nous avons utilisé l'ARC pour produire des règles d'association explicites. Ces dernières nous ont permis de mettre en relation des caractéristiques sur des pièces manufacturées et les problèmes de production. De ces règles ont pu être expliqués certains comportement menant aux erreurs du système de

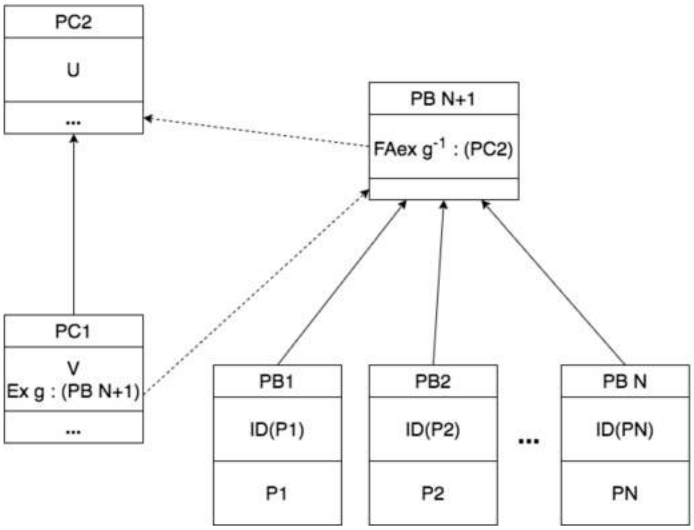


Figure 5.4: Schéma traduisant la situation de la règle #4. Les concepts PC indiquent les concepts des pièces, PB ceux des problèmes. Les flèches pleines marquent la relation d'ordre dans les treillis, les pointillés les reférences par attribut relationnel. PB N+1 porte exclusivement un attribut *relationnel* et donc ne peut être un concept-objet : il regroupe donc plusieurs problèmes. Ces problèmes co-occurrent avec des pièces de PC2.

production ainsi que déceler des insuffisances dans la modélisation. Il est envisagé, dans des travaux futurs, d'utiliser les *pattern structures*, présentées dans (Ganter et Kuznetsov, 2001), pour déterminer automatiquement les intervalles pertinents plutôt que de procéder à une binarisation a priori.

CHAPITRE VI

ANCRAGE SYMBOLIQUE ET LEXIQUE MENTAL

Si on essaye d'apprendre une nouvelle langue, utilisant uniquement le dictionnaire de celle-ci, il est nécessaire d'identifier dans un premier temps un ensemble de mots dans le dictionnaire qu'on peut rattacher à notre langue maternelle (Harnad, 1990). De plus, il faut s'assurer que cet ensemble de mots recouvre les primitives lexicales (en supposant qu'elles existent) de la langue étrangère, c'est-à-dire l'ensemble de mots permettant de définir tous les autres mots du langage. L'opération d'acquisition des premiers mots du dictionnaire étranger est appelé *problème d'ancrage symbolique*, en anglais *symbol grounding problem* (Harnad, 1990). La tâche est d'autant plus ardue lorsque les alphabets ne correspondent pas, comme le mandarin et le français. Elle est néanmoins réalisable, et même caractéristique du travail des cryptographes travaillant sur la compréhension de langues mortes (Harnad, 1990).

Pour identifier les primitives lexicales d'un langage, les travaux de (Blondin Massé *et al.*, 2008; Vincent-Lamarre *et al.*, 2016) utilisent la jointure d'ensembles d'ancrage minimaux (EAM) de dictionnaires réduits, c'est-à-dire les ensembles, comportant le plus petit nombre de mots d'un dictionnaire à partir desquels on peut définir tous les mots de ce dictionnaire. Toutefois, pour mieux caractériser ces EAM, il convient de comprendre tant leur aspect structurel que leur description

d'un point de vue psycholinguistique[1] (Vincent-Lamarre *et al.*, 2016).

Ce chapitre propose une analyse d'un ensemble de petits dictionnaires, produits par des joueurs humains. Il met en avant les co-occurences entre éléments structurels et psycholinguistiques au travers de règles d'association extraites par ARC. Dans un premier temps, la section 6.1 détaille le contexte de l'étude. Ensuite, la section 6.2 introduit la formalisation d'un dictionnaire. Puis, la section 6.3 présente un jeu sérieux, appelé «jeu du dictionnaire», qui approvisionne nos données. Finalement, la section 6.4 décrit les caractéristiques du jeu de données ainsi que l'expérience menée.

6.1 Contexte psycholinguistique

Dans la théorie du langage, tout comme dans les sciences cognitives, la recherche des primitives lexicales est un sujet très actif depuis plusieurs dizaines d'années (Goddard et Wierzbicka, 1994; Taddeo et Floridi, 2005). Ces primitives forment un ensemble de mots tel que tout mot du langage peut être défini uniquement par celles-ci. En pratique, si on intègre, de manière itérative, les définitions que l'on peut produire à partir de cet ensemble de mots, on définit tous les mots d'un dictionnaire. Dans un contexte idéal, l'ensemble des primitives lexicales devrait à la fois contenir le moins de mots possibles et être le plus expressif possible.

Une des célèbres premières initiatives, pour déterminer l'ensemble minimal de primitives lexicales, a été introduite en 1930 dans (Ogden, 1930). Bien qu'Ogden n'ait pas réussi à en construire un langage universel, on peut toujours trouver sa liste de mots dans (Ogden, 2018). Sa structure de graphe a encore été étudiée récemment (Garrido et Gutierrez, 2016), soulignant l'aspect intemporel de cette

1. la psycholinguistique est la discipline qui étudie les processus cognitifs relatifs tant à la compréhension qu'à la production du langage

ligne de recherche.

En 1972, Wierzbicka a proposé 14 primitives sémantiques, qu'elle considère comme universelles. Elles sont répertoriées dans son livre *Semantic Primitives* (Wierzbicka, 1972). Poursuivant son approche deux décennies durant, elle a étendu sa liste de mots à plus de 50 primitives sémantiques et a montré que celles-ci pouvaient être traduites dans un grand nombre de langues (Wierzbicka, 1996).

L'application *Minimal English* se base sur cette théorie (Goddard, 2018). Elle a pour but de construire manuellement un ensemble de mots de base en anglais qui permettent à la fois de décrire un grand nombre de mots plus complexes et qui sont traductibles dans de nombreuses langues différentes (Goddard, 2018). Cela permettrait, entre autres, de pourvoir la langue d'un point d'entrée efficace pour l'apprentissage de l'anglais comme langue non maternelle.

Récemment, Browne et al. (Browne, 2013; Browne, 2014) ont construit plusieurs listes «générales» de mots, telles que la *New General Service List*. Ces listes de mots, choisies avec soin pour satisfaire aux exigences des primitives lexicales, ont été principalement utilisées pour enseigner l'anglais, mais ont également été employées dans d'autres contextes (Browne *et al.*, 2013).

Bien qu'utiles en pratique, toutes ces approches commencent par proposer un ensemble de primitives et essayent, à partir de celles-ci, de construire autant de concepts que possible. Toutefois, les auteurs de ces méthodes mettent l'accent sur le fait qu'elles ne doivent pas être considérées comme complètes et définitives (Wierzbicka, 2020). Par exemple, la liste de Wierzbicka de 65 primitives inclut les concepts NOT, GOOD et BAD. Il serait possible d'argumenter que parmi les concepts GOOD et BAD, l'un d'entre eux, au moins, pourrait être retiré, puisque chacun peut être défini en combinant l'autre avec NOT c'est-à-dire GOOD = NOT BAD et BAD = NOT GOOD.

Toujours pour répondre au problème d'ancrage symbolique, une approche complémentaire a été développée par Blondin Massé et al. (Blondin Massé *et al.*, 2008). À la place de démarrer par un ensemble fixé de mots, l'article soulève la possibilité de considérer conjointement les ensembles d'ancrages minimaux (EAM) extraits de multiples dictionnaires. Ces EAM sont des ensembles de mots suffisants pour définir tous les autres. Bien que le nombre de EAM d'un dictionnaire puisse être exponentiel par rapport au nombre de mots, tous les EAM semblent partager des caractéristiques psycholinguistiques communes (Vincent-Lamarre *et al.*, 2016). En effet, il a été montré que les mots de EAM communs à différents dictionnaires sont des mots utilisés fréquemment, appris jeunes et concrets (Vincent-Lamarre *et al.*, 2016).

Pour ces variables psycholinguistiques (âge d'acquisition, fréquence, concrétude), on peut également noter une différence, selon la partie du dictionnaire dans laquelle un mot se trouve. Si on représente un dictionnaire comme un graphe, tel que décrit dans la section 6.2, où les mots sont les sommets et les arcs matérialisent la relation de définition, alors le *coeur* est la plus grande composante fortement connexe (CFC) et les *satellites* sont les CFCs dont les circuits sont plus courts. Or, il a été précédemment observé que, dans un EAM, les mots du coeur sont plus fréquents, plus abstraits et appris plus tôt, à l'inverse des mots des satellites (Vincent-Lamarre *et al.*, 2016).

6.2 Formalisation

Dans cette section, nous introduisons le formalisme mathématique qui permet de modéliser les lexiques et les graphes, afin d'en étudier l'aspect structurel.

6.2.1 Lexiques

La formalisation des lexiques est adaptée de (Poulin *et al.*, 2018). Commençons par les définitions des termes principaux.

Définition 16. *(Poulin* et al.*, 2018) Un* lexique complet non-ambigu *est un quadruplet* $X = (\mathcal{A}, \mathcal{P}, \mathcal{L}, \mathcal{D})$ *où*

- $\mathcal{A}$ *est un ensemble fini appelé un* alphabet*, dont les éléments sont appelés des* lettres *;*
- $\mathcal{P}$ *est un ensemble fini dont les éléments sont appelés* catégories grammaticales *;*
- $\mathcal{L}$ *est un ensemble fini de triplets de la forme* $\ell = (w, i, p)$*, appelés* lexèmes *et dénotés par* $\ell = w_p^i$*, où* $w \in A^*$ *est un mot,* $i \geq 1$ *est un entier et* $p \in P$*. On dit alors que* (w, i, p) *est le* i-ème sens *du* mot étiqueté (w, p)
- $\mathcal{D}$ *est une application partielle qui associe à un lexème* $\ell \in \mathcal{L}$ *une séquence finie et non vide* $D(\ell) = (\ell_1, \ell_2, \ldots, \ell_k)$*, où pour tout* i*,* $d_i \in \mathcal{L}$*. Une telle séquence s'appelle la* définition *de* ℓ*.*

À ces termes s'appliquent les contraintes suivantes :

lexème fonctionnel *L'ensemble* $\mathcal{P}$ *contient un élément spécial appelé* stop *et dénoté* S*, identifiant les* lexèmes fonctionnels *(en anglais* stop words*) ;*

complétude *Pour chaque* $(w, i, p) \in \mathcal{L}$*, si* $p \neq S$*, alors* $\mathcal{D}(w, i, p)$ *est bien défini.*

numérotation cohérente des lexèmes *Si* $(w, i, p) \in \mathcal{L}$ *et* $i > 1$ *alors* $(w, i - 1, p) \in \mathcal{L}$ *;*

Si pour tout $(w, i, p) \in \mathcal{L}$, on a $i = 1$, alors le lexique $X = (\mathcal{A}, \mathcal{P}, \mathcal{L}, \mathcal{D})$ est qualifié de *monosémique*. Dans de tels cas on écrit simplement w_p au lieu de w_p^i.

Autrement dit, un lexique complet non-ambigu est une liste de lexèmes qui sont tous définis, à part les lexèmes fonctionnels, et où toutes les définitions sont non-

ℓ	$D(\ell)$
BIG_A	$(\text{NOT}_S, \text{SMALL}_A)$
HUGE_A	$(\text{VERY}_S, \text{BIG}_A)$
SMALL_A	$(\text{NOT}_S, \text{BIG}_A)$

Tableau 6.1: Exemple de lexique complet non-ambigu.

ambigües.

Prenons un exemple :

Exemple 34. *Soit* $X = (\mathcal{A}, \mathcal{P}, \mathcal{L}, \mathcal{D})$*, où* $\mathcal{A} = \{a, b, \ldots, z\}$*,* $\mathcal{P} = \{A, S\}$ *(*A *= adjectif,* S *= stop) et* $\mathcal{L}$*,* $\mathcal{D}$ *sont définis dans le tableau 6.1. Chaque lexème utilisé dans une définition est lui-même défini, à l'exception des lexèmes fonctionnels* NOT_S *et* VERY_S*. Alors, le lexique* X *est complet et non-ambigu.*

En pratique, les mots référencés avec S sont ceux jouant simplement un rôle syntaxique et n'ont que très peu de valeur sur le plan sémantique (par exemple, *no*, *the*, *a*). Toutefois, rien n'empêche d'affecter arbitrairement un mot à cette catégorie, au besoin.

Pour la suite, on fixe $\mathcal{P} = \{N, V, A, R, S\}$, correspondant aux catégories grammaticales *nom*, *verbe*, *adjectif*, *adverbe* et *stop*.

6.2.2 Graphes

Un *graphe orienté* est un couple $G = (V, A)$, où V est un ensemble fini d'éléments appelés *nœuds* et $A \subseteq V \times V$ est un ensemble fini d'éléments appelés *arcs*.

La *densité* de G, notée density(G), est la proportion d'arcs présents dans le graphe par rapport à la quantité d'arcs possibles, c'est-à-dire density(G) $= |A|/|V|^2$.

Soient un graphe $G = (V, A)$, deux nœuds $u, v \in V$ et un entier positif k. On

dit que $p = (v_1, v_2, \ldots, v_k)$ est un *uv-chemin (orienté)* de G si $u = v_1$, $v = v_k$ et $(v_i, v_{i+1}) \in E$ pour $i = 1, 2, \ldots, k-1$. En particulier, si $u = v$, le chemin p est appelé un *circuit* de G.

Soient deux nœuds u et v. On dénote l'existence d'un uv-chemin dans G par $u \rightarrow_G v$, ou, plus simplement, par $u \rightarrow v$ si le graphe G est clair dans le contexte exprimé. De plus, on écrit $u \leftrightarrow v$ si et seulement si $u \rightarrow v$ et $v \rightarrow u$. La relation $\leftrightarrow$ étant réflexive, symétrique et transitive, elle est donc une relation d'équivalence.

Une classe d'équivalence selon la relation $\leftrightarrow$ est appelée *composante fortement connexe (CFC)* de $G = (V, A)$. Elle représente un sous-ensemble de nœuds de V tel que, pour toute paire de nœuds de la composante, il existe des chemins orientés reliant ces deux nœuds dans les deux directions. Les CFC d'un graphe peuvent être calculées à l'aide de différents algorithmes tels que l'algorithme de Tarjan (Tarjan, 1972).

Il est plus pratique, pour plusieurs métriques, de considérer la version non orientée d'un graphe orienté. Si l'on prend deux nœuds u, v d'un graphe orienté $G = (V, A)$ et un entier positif k, on dit que $p = (v_1, v_2, \ldots, v_k)$ est une *uv-chaîne* de G si $u = v_1$, $v = v_k$ et que, pour chaque $i = 1, 2, \ldots, k-1$, on ait $(v_i, v_{i+1}) \in E$ ou $(v_{i+1}, v_i) \in E$. La *longueur* d'une uv-chaîne, notée $|p|$, est le nombre $k-1$, c'est-à-dire le nombre d'arcs parcourus par p. La *distance entre u et v*, notée $\mathrm{dist}(u, v)$, est la longueur de la plus petite chaîne entre u et v, c'est-à-dire

$$\mathrm{dist}(u, v) = \min\{|p| : p \text{ est une } uv\text{-chaîne}\}.$$

De ces définitions, on peut extraire des statistiques structurales pour un graphe G donné. Par exemple, le *diamètre de G*, noté $\mathrm{diam}(G)$, est la distance maximale

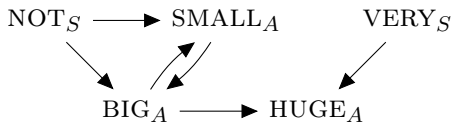


Figure 6.1: Le graphe orienté du lexique complet non-ambigu de l'exemple 34.

qu'on peut trouver entre deux nœuds de G. Formellement,

$$\mathrm{diam}(G) = \max\{\mathrm{dist}(u, v) : u, v \in V\}.$$

Enfin, la *longueur de chaîne ou chemin charactéristique* de G représente la longueur moyenne du plus court chemin entre deux sommets. Elle est notée $\mathrm{CPL}(G)$ et est égale à

$$\mathrm{CPL}(G) = \sum_{u,v \in V} \frac{\mathrm{dist}(u, v)}{|V|(|V| - 1)}.$$

Si $X = (\mathcal{A}, \mathcal{P}, \mathcal{L}, \mathcal{D})$ est un lexique complet non-ambigu, alors le *graphe* de X est le graphe orienté $\mathrm{Graph}(X) = (\mathcal{L}, A)$, tel que $(\ell_1, \ell_2) \in A$ si, et seulement si, le lexème ℓ_1 apparaît dans la définition $\mathcal{D}(\ell_2)$ du lexème ℓ_2. Le graphe du lexique décrit dans l'exemple 34 est représenté en figure 6.1. En d'autres termes, le graphe illustre la relation «apparaît dans la définition» sur l'ensemble des lexèmes.

6.3 Le *jeu du dictionnaire*

Dans le but de mieux comprendre l'origine des variations psycho-linguistiques sur les différentes parties des EAM, un jeu sérieux, appelé *jeu du dictionnaire*, a été développé (Vincent-Lamarre *et al.*, 2017). C'est un jeu en ligne, ouvert et accessible à tous, dont le but est de créer des dictionnaires complets en anglais (Picard *et al.*, 2013; Vincent-Lamarre *et al.*, 2016) (une version en français est aussi disponible, mais, par manque de ressources psycholinguistiques, nous n'en avons pas

exploité les données). Sur la page http://lexis.uqam.ca:8080/dictGame/index.jsp, on trouve une description complète du jeu et on peut créer son propre dictionnaire (Vincent-Lamarre *et al.*, 2017). L'objectif des joueurs est de créer les structures les plus petites possibles. Au début d'une partie, un joueur commence par choisir un mot de départ fixé (appelé mot *racine*) dont il donne une définition. Dans la version actuelle, quatre mots racines sont proposés : *clock*, *horse*, *person*, *thing*. Ensuite, il se retrouve face à l'écran présenté en figure 6.2. Il doit, un à un, définir tous les mots utilisés pour définir le mot racine. Puis, il doit définir les mots qui servent dans les définitions des mots qui définissent le mot racine. Récursivement, chaque mot utilisé dans une définition doit à son tour être défini. Bien évidemment, un mot peut servir dans de multiples définitions. Toutefois, une seule définition est demandée pour chaque mot, il n'est pas nécessaire de redéfinir un mot chaque fois qu'il apparaît dans une définition. La partie prend fin lorsque tous les mots qui ont été employés dans une définition sont définis, à l'exception des lexèmes fonctionnels. On qualifie alors le dictionnaire de *fermé* (Picard *et al.*, 2013; Vincent-Lamarre *et al.*, 2016).

Les auteurs de (Vincent-Lamarre *et al.*, 2017) considèrent comme lexème fonctionnel :

- tout mot n'étant ni un verbe, ni un adverbe, ni un adjectif, ni un nom commun (ex : déterminants, pronoms ...) ,
- le verbe *to be* et toutes ses formes conjuguées,
- les adverbes *no* et *not*.

Finalement, on ajoute une contrainte supplémentaire pour forcer un minimum d'expressivité dans les dictionnaires produits : toute définition doit contenir au moins trois lexèmes qui ne soient pas des lexèmes fonctionnels. Si cette contrainte n'est pas satisfaite, un avertissement apparaît, invitant le joueur à corriger sa définition.

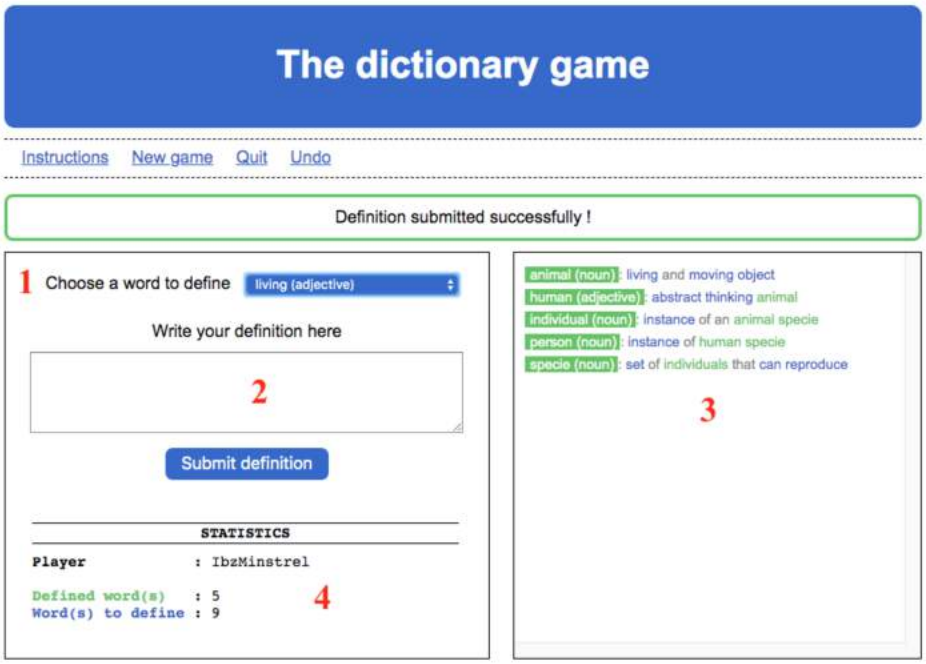


Figure 6.2: Capture d'écran du jeu des dictionnaires (Vincent-Lamarre *et al.*, 2017). La zone 1 présente un menu déroulant où le joueur peut sélectionner un mot et produire sa définition dans la zone 2. La zone 3 montre toutes les définitions produites par le joueur. Les mots écrits en vert sont définis alors que ceux en noir non. Les mots en gris sont les lexèmes fonctionnels. La zone 4 présente la progression du joueur.

L'exemple 35 souligne quelques définitions qui pourraient être utilisées pour démarrer la construction d'un nouveau dictionnaire.

Exemple 35. *Dictionnaire partant du mot-racine* horse *:*

— horse *: animal on which one human rides*

— animal *: organism that belongs to the living kingdom*

— human *: animal species that possess reason*

— *etc.*

Le résultat obtenu en fin de partie est un dictionnaire fermé qui vérifie l'ensemble des propriétés permettant de le qualifier de lexique complet non-ambigu (cf. définition 16).

Chaque dictionnaire produit et récupéré a ensuite été adapté en graphe orienté, en utilisant la transformation naturelle, présentée en section 6.2. La figure 6.3 présente la représentation en graphe d'un dictionnaire extrait par le jeu (pour des raisons de lisibilité, nous avons représenté le plus petit dictionnaire produit par un joueur).

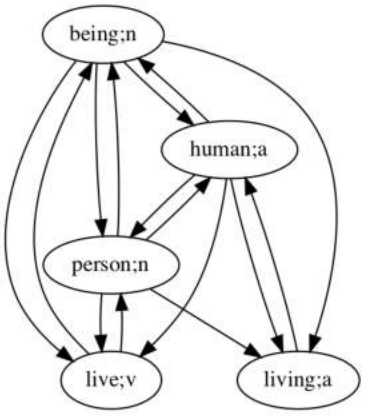


Figure 6.3: Graphe représentant un dictionnaire de cinq mots : *being*, *humain*, *live*, *living* et *person*

6.4 Analyse relationnelle

Cette section présente notre approche ainsi que les résultats obtenus. On commence par présenter les données récupérées au travers du jeu du dictionnaire, puis on détaille la modélisation adoptée, avant de présenter les résultats de l'analyse.

6.4.1 Données

Nos données, hétérogènes, sont constituées des graphes des 60 dictionnaires récupérés par le jeu, ainsi que de caractéristiques psycholinguistiques sur les 2525 mots employés dans les dictionnaires.

Pour observer la structure sous-jacente des dictionnaires, les statistiques suivantes

	numV	*numE*	*nSCC*	*CPL*	*dens*	*diam*
Mean	153.4	542.2	8.1	5.3	0.026	14.5
Std	72.8	275.7	13.5	1.3	0.008	4.5
Min	91.0	309.0	1.0	2.0	0.008	9.0
Max	433.0	1558.0	88.0	11.4	0.041	35.0

Tableau 6.2: Caractéristiques des graphes relatifs aux dictionnaires

ont été calculées pour chacun des graphes correspondant à un dictionnaire *fermé* produit par un joueur :

— *numV* : nombre de sommets,

— *numE* : nombre d'arcs,

— *nSCC* : nombre de composantes fortement connexes,

— *CPL* : longueur de chaîne caractéristique,

— *dens* : densité,

— *diam* : diamètre.

Après avoir évalué le nombre de sommets d'un dictionnaire, nous avons retiré les dictionnaires de taille aberrante : quelques dictionnaires présentaient moins de 35 mots ; après leur retrait, le plus petit dictionnaire comptait 91 mots. Après filtrage, notre jeu de données est constitué de 60 dictionnaires. Le tableau 6.4 présente la répartition de ces dictionnaires selon le mot racine choisi par les joueurs. La distribution des dictionnaires selon les variables structurelles (*numV*, *numE*, *nSCC*, *CPL*, *dens*, *diam*) est donnée par le tableau 6.2.

L'objectif étant d'évaluer les relations entre les données structurelles des dictionnaires et les aspects psycholinguistiques des mots qu'ils contiennent, il nous reste donc à introduire les normes évaluant les mots des dictionnaires. Dans le tableau 6.3, on trouve un extrait du tableau détaillant les propriétés psycholinguistiques d'un mot, à savoir :

Age d'acquisition : La valeur AOAB d'un mot, repertoriée dans (Kuperman

	AOAB	AOAC	*Conc*	*FreqP*	*FreqL*
abandon	8.32	—	2.54	8.10	1
abide	9.50	4.00	1.68	2.71	1
ability	8.84	—	1.81	19.22	25
able	7.79	4.77	2.38	159.90	30
absence	7.70	—	2.31	6.31	2
absent	6.50	8.28	2.70	2.57	1
...					

Tableau 6.3: Extrait de la table de propriétés psycholinguistiques.

Racine	Nombre de dictionnaires
clock	22
horse	19
person	10
thing	9
Total	60

Tableau 6.4: Nombre de dictionnaires par mot racine

et al., 2012), présente une estimation de l'âge où celui-ci est acquis par un individu, selon la norme de Brysbaert. La norme définie par Childes définit la valeur d'âge d'acquisition AOAC d'un mot. Les valeurs sont répertoriées dans (MacWhinney, 2000).

Concrétude : La variable *Conc*, sur une échelle de 1 à 5, évalue si un mot est abstrait ou concret (Brysbaert *et al.*, 2014).

Fréquence : *FreqP* – (Brysbaert et New, 2009) est une mesure du taux d'occurrence d'un mot dans le corpus SUBTLEX$_{US}$[2]. La fréquence locale *FreqL*, quant à elle, est une mesure de la fréquence du mot dans le corpus constitué exclusivement des dictionnaires récupérés par le jeu du dictionnaire : c'est le ratio du nombre de définitions contenant un certain mot par rapport au nombre total de toutes les définitions produites dans tous les dictionnaires.

Le symbole '—' représente une donnée non disponible.

2. Ce corpus de 51 millions de mots englobe les sous-titres de 5000 épisodes de séries de 5000 films américains.

Dictionnaires			Mots		
Propriété	Valeur	Catégorie	Propriété	Valeur	Catégorie
numV	[91, 107]	lo	AOAB	[2.3, 6.8]	young
	[108, 152]	med		[6.9, 11.4]	middle
	[153, 433]	hi		[11.5, 16.2]	older
numE	[309, 725]	lo	AOAC	[1.3, 4.4]	young
	[726, 1141]	med		[4.5, 7.6]	middle
	[1142, 1158]	hi		[7.7, 11.0]	older
nSCC	[1, 29]	lo	CONC	[1.1, 2.3]	lo
	[30, 59]	med		[2.4, 3.6]	med
	[60, 88]	hi		[3.7, 5.0]	hi
CPL	[2, 5.1]	lo	FREQP	[0.0, 3.9]	lo
	[5.2, 8.3]	med		[4.0, 27.8]	med
	[8.4, 11.4]	hi		[27.9, 6161.5]	hi
dens	[0.008, 0.018]	lo	FREQL	[0, 1]	Q1
	[0.019, 0.03]	med		[2, 3]	Q2
	[0.031, 0.041]	hi		[4, 7]	Q3
				[8, 58]	Q4
diam	[9, 17.6]	lo			
	[17.7, 26.2]	med			
	[26.3, 35]	hi			

Tableau 6.5: Discrétisation des variables structurelles et psycholinguistiques

6.4.2 Modélisation

Que ce soit les variables psycholinguistiques des mots ou structurelles des dictionnaires, l'ARC ne peut les traiter avant une étape de discrétisation. Le tableau 6.5 présente les transformations que nous avons appliquées aux variables : les variables structurelles sur les graphes, les âges d'acquisition et la concrétude sont discrétisés sur trois intervalles égaux ; les fréquences suivent par contre une répartition uniforme, c'est-à-dire que les intervalles ne sont pas nécessairement de taille égale, mais de cardinalité égale, c'est-à-dire qu'ils sont calculés pour qu'il y ait autant de mots dans chaque catégorie.

Puisque le jeu du dictionnaire a pour but de faire créer aux joueurs des dictionnaires les plus petits possibles (nombre minimal de mots), nous nous sommes

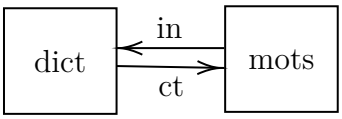


Figure 6.4: Schéma de la RCF du *jeu du dictionnaire*

intéressés aux stratégies employées pour «gagner». Pour cela, nous avons inspecté les attributs structurels des lexiques produits, conjointement avec une description psycholinguistique des mots utilisés dans les dictionnaires.

Pour extraire les associations entre les caractéristiques des mots et des dictionnaires, nous avons construit une famille relationnelle de contexte (FRC) conforme au schéma présenté en figure 6.4. Le contexte *dict* décrit les dictionnaires produits. Les attributs sont, d'une part, les variables catégorielles présentées au tableau 6.5 (à savoir les métriques issues de la théorie des graphes présentées) et, d'autre part, les quatre mots racines permettant de signifier l'origine du dictionnaire.

Le contexte *mots* décrit tous les mots présents dans au moins un dictionnaire. Les attributs de ce contexte sont les catégories discrétisées des propriétés psycholinguistiques présentées dans le tableau 6.5.

Enfin, nous avons également inclus une relation objet-à-objet *in*, et son inverse *ct* ; reflétant l'utilisation d'un mot dans un dictionnaire. Nous avons spécialement conçu et utilisé les opérateurs de graduation $\cap_{>p\%}$. Rappelons que, si $p = 30$, alors l'attribut $\cap_{>30\%} in : (C_1)$ pour un mot signifie qu'au moins 30% des dictionnaires dans lesquels ce mot est inclus sont dans l'extension du concept C_1. On note que $\cap_{\geq p\%}$ est une version relaxée de l'opérateur $\forall\exists$.

#	Prémisse	Conclusion	Support Prémisse	Support Règle	Confiance
1	$\forall\exists\ in : numV(\text{hi})$	$\text{FrL}(q_1)$	35%	27%	78%
2	person, $\cap_{>60\%}\ ct : \text{FrL}(q_4)$	$numV(\text{lo})$	6.6%	5%	75%
3	$numV(\text{lo})$	$\cap_{>50\%}\ ct : \text{FrL}(q_4)$	30%	27%	89%
4	$numV(\text{hi})$, person	$\cap_{>50\%} ct : \text{FrL}(q_4)$	6.6%	6.6%	100%
5	$Conc(\text{med}), \cap_{>70\%} in : (numV(\text{lo}))$	AOAB(med)	2.3%	1.7%	76%
6	$numV(\text{lo})$	$dens(\text{hi})$	30%	27%	89%
7	$numV(\text{hi})$	lCFC(hi)	33%	28%	85%
8	$Conc(\text{lo}), \forall\exists\ in : numV(\text{lo})$	$\text{FrL}(q_1)$	3.4%	3.1%	92.3%
9	$Conc(\text{med}), \cap_{>50\%}\ in : numV(\text{lo})$	AOAB(med)	2.4%	1.8%	75%
10	AoaB(young), AoaC(young), $\cap_{>40\%}$ in $numV(\text{hi})$ $\cap_{>70\%}$ in (horse)	$Conc(\text{hi})$	2.6%	2.1%	79%

Tableau 6.6: Extrait des règles d'associations produites de la RCF figure 6.4

6.4.3 Résultats

Nous présentons maintenant les résultats de notre expérience. Ceux-ci se présentent sous la forme de règles d'association extraites par l'ARC après une itération unique.

En utilisant les résultats de ARC après une seule itération, on découvre des observations intéressantes, tant sur les mots (24 954 règles) que sur les dictionnaires (206 476 règles). Certaines de ces règles sont présentées dans le tableau 6.6. La première colonne indique le numéro de règle pour reporter les interprétations plus bas. La prémisse et la conclusion d'une règle sont respectivement les parties à gauche et à droite du symbole $\rightarrow$ d'une règle d'association. Soit une règle $r : U \rightarrow V$ pour un contexte formel $K = (O, A, I)$ (ici, soit des mots soit des dictionnaires). Le *support de la prémisse*, le *support de la règle* et la *confiance de la règle*, respectivement notées $sp(r)$, $sr(r)$ et $c(r)$ sont données par

$$sp(r) = \frac{|U'|}{|O|} \tag{6.1}$$

$$sr(r) = \frac{|(U \cup V)'|}{|O|} \tag{6.2}$$

$$c(r) = \frac{|(U \cup V)'|}{|U'|} \tag{6.3}$$

Il est à noter qu'après avoir produit les règles, nous nous sommes concentrés sur celles présentant de l'information relative à la taille des dictionnaires. Présentons maintenant notre interprétation des règles du tableau 6.6.

#1 35% des mots sont employés uniquement dans des grands dictionnaires, et 78% de ces derniers (soit 27% de tous les mots), sont peu employés par nos joueurs. On pourrait supposer que les grands dictionnaires utilisent beaucoup de chemins non-optimaux pour définir leurs termes, créant ce que l'on pourrait qualifier de «bruit».

#2 75% des dictionnaires de racine *person* contenant principalement (à plus de 60%) des mots utilisés très fréquemment dans le jeu du dictionnaire sont petits. De manière duale à la règle précédente, celle-ci semble indiquer qu'un «graphe optimal» peut exister, et que les dictionnaires tendent à converger vers cet optimal.

#3 89% des petits dictionnaires ont plus de la moitié de leurs mots qui sont très fréquemment utilisés par nos joueurs. Cette troisième interprétation est cohérente avec les précédentes.

#4 Dans tous les grands dictionnaires de racine *person*, plus de la moitié des mots sont utilisés très fréquemment par les joueurs. Ce résultat peut sembler contradictoire avec les trois premières observations. Toutefois, il est aussi envisageable que la racine joue un rôle crucial dans cette observation ; ou alors, on pourrait également imaginer qu'on ait bien 50% des mots qui soient

dans la direction d'un optimal mais que ce sont les autres qui créent la plus grande partie du bruit.

Mises à part ces observations, l'analyse de différents sous-ensembles de règles nous fournit également des co-occurences supplémentaires à propos du jeu :

#5 Parmi l'ensemble des mots qui sont de concrétude moyenne et que l'on peut trouver principalement dans des petits dictionnaires, 76% ont également un âge d'acquisition moyen.

#6 Les petits dictionnaires sont très denses.

#7 Les plus grands dictionnaires ont aussi de larges composantes fortement connexes.

#8 Parmi l'ensemble des mots très abstraits qui n'existent que dans les petits dictionnaires, 92% sont rarement utilisés par les joueurs.

#9 Parmi l'ensemble des mots qui ne sont ni très concrets ni très abstraits, que l'on retrouve principalement dans les petits dictionnaires, 75% sont appris à âge moyen.

#10 Parmi l'ensemble des mots appris jeunes (selon les mesures de Childes et Brysbaert) et utilisés principalement dans les dictionnaires de mot racine *horse*, 79% sont aussi des mots très concrets.

Les hypothèses que nous pouvons tirer de ces observations sont les suivantes : les petits dictionnaires tendent vers un ensemble «universel» de mots. La plupart des joueurs convergent de manière plus ou moins complète vers cet ensemble. Caractériser au mieux cet ensemble semble donc critique pour l'analyse des dictionnaires produits via le jeu.

Toutefois, une large partie de l'ensemble universel apparaît aussi dans les grands dictionnaires. Dans le second cas, il y a une plus grande variabilité à l'intérieur des dictionnaires.

Le scénario latent semble être le suivant : pour un mot racine donné, il y a un ensemble universel de définitions à sélectionner. Mais pour un joueur, certaines de ces définitions ne sont pas simples à formuler. Lorsqu'un joueur est hésitant sur une définition, celle-ci perd en concision, et un ensemble de définitions plus important doit être produit pour compenser l'absence d'un mot.

Ces déviations n'ont aucune raison d'arriver aux mêmes mots pour des joueurs différents. C'est de cette manière que se formeraient les mots rares, spécifiques à chaque joueur.

De plus, les petits dictionnaires sont, comme on peut le supposer, très denses. Cela peut se comprendre par le fait que la stratégie pour produire de petits dictionnaires est de réutiliser les mots dans un maximum de définitions possibles. Cela devrait avoir pour effet d'augmenter le nombre d'arcs sans augmenter le nombre de sommets. À l'inverse, les grands dictionnaires ont une densité faible à moyenne.

6.5 Conclusion

La forme explicite et interprétable des règles d'association extraites nous permet de formuler plusieurs hypothèses en exposant les co-occurences entre les caractéristiques structurelles des dictionnaires et les mesures psycholinguistiques des mots les composant. Notamment, ces règles soulèvent certaines lacunes dans la modélisation.

Tout d'abord, la variable $numV$ semble être une bonne heuristique de départ pour mesurer la qualité d'une stratégie, mais elle ne prend en compte ni la concision, ni la précision, ni la pertinence des définitions. De plus, le mot racine semble influencer significativement les résultats et, par conséquent, dans une seconde expérience, nous projetons d'établir les catégories structurelles non plus en absolu,

mais relativement à la racine. Il conviendrait aussi de vérifier si le mot racine est, par lui-même, d'importance, ou si c'est en réalité sa catégorie psycholinguistique qui influence la structure du dictionnaire.

Ainsi, plusieurs pistes de recherche restent ouvertes :

— intégrer non pas simplement les mots, mais aussi les définitions ;

— adapter les catégories structurelles selon les racines ;

— intégrer les attributs psycholinguistiques des racines.

— déterminer les caractéristiques des mots spécifiques aux petits dictionnaires

De plus, il pourrait être intéressant de comparer, les résultats à ceux que produiraient un lexicologue. Bien évidemment, même si les règles d'association permettent l'analyse du jeu de données, il est nécessaire d'en augmenter la taille pour vérifier si nos hypothèses, basées sur les règles de haute confiance mais de faible support (comme les règles #2 et #4), sont généralisables ou relèvent de cas particuliers.

CHAPITRE VII

INGÉNIERIE DE CONNAISSANCES PAR ANALYSE DE CONCEPTS

Dans ce chapitre, nous présentons trois différentes sous-tâches de l'ingénierie de la connaissance auxquelles l'ARC peut contribuer. Pour les illustrer, nous utilisons un jeu de données artificiel, et montrons les avantages de la suppression des définitions circulaires en confrontant les résultats ainsi obtenus à ceux produits par les méthodologies de la littérature.

7.1 Introduction

L'ingénierie de connaissance, comme on l'a vu au chapitre 1, est une discipline qui vise à concevoir une base de connaissance, puis à créer un agent qui peut la manipuler pour répondre à certaines tâches (Russell et Norvig, 2002). On s'intéresse ici à la première partie : la conception de la base de connaissance.

On modélise ici les jeux de données au travers de diagrammes de classe. Ces derniers sont des schémas en langage de modélisation unifié, en anglais *unified modeling language*(*UML*) permettant de représenter le type d'un ensemble d'objets dans une classe et d'établir les connexions sémantiques entre deux classes au travers d'associations (Berardi *et al.*, 2005). La figure 7.4 présente un exemple. Chaque «boîte» représente le type d'un ensemble d'objets, dont le nom est donné dans la partie supérieure. Une telle «boîte» est appelée *classe*. On observe par

exemple la classe `EntitéLégale` qui décrit toutes les instances étant des *entités légales*. La seconde partie de la «boîte» présente les *attributs* de la classe qui définissent celle-ci. Par exemple, une *entité légale* est un objet ayant un *nom* et une *adresse*. Une «flèche pleine», comme celle reliant la classe `Personne` à la classe `EntitéLégale`, est une *relation d'héritage*. Elle stipule que tout objet de la classe *Personne* est aussi dans la classe `EntitéLégale`. Du fait de cette inclusion, les *attributs* de la classe parente ne sont pas reportés dans la sous-classe, c'est-à-dire que la classe `Personne` porte tout autant l'attribut *date de naissance* que *nom* et *adresse*. Un «trait plein» représente une *association* entre deux classes, c'est-à-dire une connexion sémantique telle que le lien entre un *musicien* et ses *instruments* qui caractérise la possession. Une *association* est un ensemble de couples d'objets des deux classes. Une «flèche pointillée» n'entre pas dans les standards *UML* mais représente ici une relation d'héritage entre deux associations et est utilisé dans les schémas d'ontologies.

Considérons un jeu de données relationnelles avec un schéma structurant ces données (le schéma le plus pauvre possible est constitué d'une classe unique décrivant tous les objets du jeu de données). Un processus d'ingénierie de connaissance peut alors être composé des trois étapes, réitérées au besoin.

— **Restructurer le schéma.** En utilisant uniquement les informations du schéma, notamment les attributs de classe, on peut découvrir de nouvelles classes, de nouvelles relations ainsi que des subsomptions entre les divers éléments.

— **Évaluer le typage.** Une fois le schéma restructuré et de nouvelles classes incorporées, les objets doivent ensuite être réétiquetés par les classes adaptées. Le typage, c'est-à-dire la correspondance entre les données et les classes, doit être évalué et les anomalies détectées.

— **Raffiner le schéma.** Après que les données soient intégrées au nouveau

schéma et vérifiées, on peut les utiliser pour suggérer de nouvelles classes et de nouvelles relations.

Ces tâches peuvent être effectuées manuellement. Les modifications sont souvent alors faites à échelle locale (par exemple, le déplacement d'une instance dans une sous-classe), ce qui peut créer des effets de bord et nuire à la qualité globale d'un schéma (Gröner et Staab, 2010). L'ARC, comme décrite aux chapitres 3 et 4, est une méthode d'analyse automatique visant à organiser les abstractions que l'on peut former à partir de données relationnelles dans une hiérarchie, qui peut servir de fondement pour la création et l'évaluation d'une base de connaissance. Elle est notamment utilisée dans différentes tâches d'ingéniérie de connaissance (Rouane-Hacene *et al.*, 2010; Rouane-Hacene *et al.*, 2011; Hacene *et al.*, 2008; David *et al.*, 2018) et d'ingénierie logicielle (cf. section 3.1). En vertu de ses capacités à manipuler les données relationnelles, à créer des règles d'association permettant d'évaluer conjointement les caractéristiques des objets et des classes, ainsi qu'à l'ordonnancement en treillis de concepts formels, l'ARC semble être un outil de choix pour intervenir aux trois niveaux du processus d'ingéniérie de la base de connaissance. Les trois prochaines sections de ce chapitre illustrent, au travers d'un exemple, comment l'ARC dans sa forme acyclique peut répondre à chacune de ces trois tâches.

7.2 Restructuration

Rouane-Hacene et al. suggèrent que l'on peut utiliser l'ARC comme outil d'analyse de la structure d'un modèle relationnel de données (Rouane-Hacene *et al.*, 2010; Rouane-Hacene *et al.*, 2011).

La méthodologie à appliquer selon ces articles est la suivante. On conçoit deux contextes formels : le premier a pour objets les classes du modèle, et en attri-

buts les variables d'instance ; le second décrit l'ensemble de toutes les associations du schéma. En plus de ces deux contextes, il faut deux relations formelles : la première, liant une association à la classe de son domaine, la seconde liant une association à sa classe codomaine. Le domaine et le codomaine de chacune de ces associations sont uniques. Basons-nous maintenant sur l'exemple du diagramme UML en figure 7.1. Il présente huit classes, parmi lesquelles les instruments de musique *piano*, *guitare* (`Guit.` sur la figure) et *guitare électrique* (`eGuit.` sur la figure), les catégories de personnes *politicien*, *musicien* et *artiste musicien* (c'est-à-dire musicien «professionnel», noté `ArtistMusic.` sur la figure), ainsi que les organisations *parti* politique et *groupe* de musique. Des attributs caractérisent ces classes, tels que le nombre de frettes d'une guitare, ou la date de naissance des individus. Finalement, des associations plusieurs-à-plusieurs entre les classes sont représentées par les lignes reliant deux classes. On compte, entre autres, parmi celles-ci les associations entre les classes *musicien* et *guitare*, entre *artiste musicien* et *guitare électrique*, *politicien* et *parti*, etc.

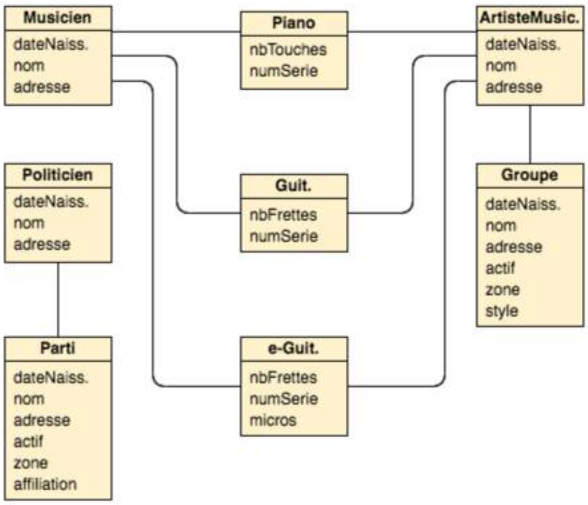


Figure 7.1: Diagramme de classes de l'exemple de restructuration

En appliquant la méthodologie proposée par Rouane-Hacene et al., le diagramme de classe de la figure 7.1 est transformé en la famille relationnelle proposée au

tableau 7.1 (Rouane-Hacene *et al.*, 2010; Rouane-Hacene *et al.*, 2011). Elle contient un contexte formel regroupant les classes, un autre les associations, une relation décrivant la classe de domaine d'une association et une relation décrivant la classe de codomaine d'une association. Dans le contexte association, on trouve les associations liant un musicien (M) ou un artiste musicien (AM) à un instrument pouvant être une guitare (G), une guitare électrique (eG) ou un piano (P). Les associations *AM-B* et *P-Pa* décrivent respectivement si un artiste musicien fait partie d'un groupe de musique, et si un politicien est membre d'un parti politique.

Une association plusieurs-à-plusieurs entre deux classes reflète le fait qu'au moins une instance d'une classe puisse être reliée à au moins une instance de l'autre classe. Partant de ce constat, l'opérateur $\exists$ semble le plus adéquat, pour la propositionnalisation du domaine et du codomaine d'une association.

Ainsi, on traite par ARC la FRC du tableau 7.1 en utilisant l'opérateur de graduation *existentiel* sur les deux relations. Au point fixe, on obtient les diagrammes de Hasse du treillis des classes et de treillis des associations illustrés dans la figure 7.2.

Bien évidemment, on trouve des circuits entre les attributs (par exemple le concept 4 du treillis des classes se réfère indirectement au travers du concept 1 du treillis des propriétés). Ces circuits rendent l'interprétation difficile. De plus, à part les circuits, la procédure proposée est complexe : on doit choisir les concepts qui définissent des classes puis après sélectionner les associations qui conviennent. Supposons que les classes pertinentes ont été extraites du treillis et intégrées au modèle. Comment sélectionner alors les associations à extraire du second treillis ? Par exemple, prenons le concept 1. Il y a plusieurs domaines et codomaines possibles.

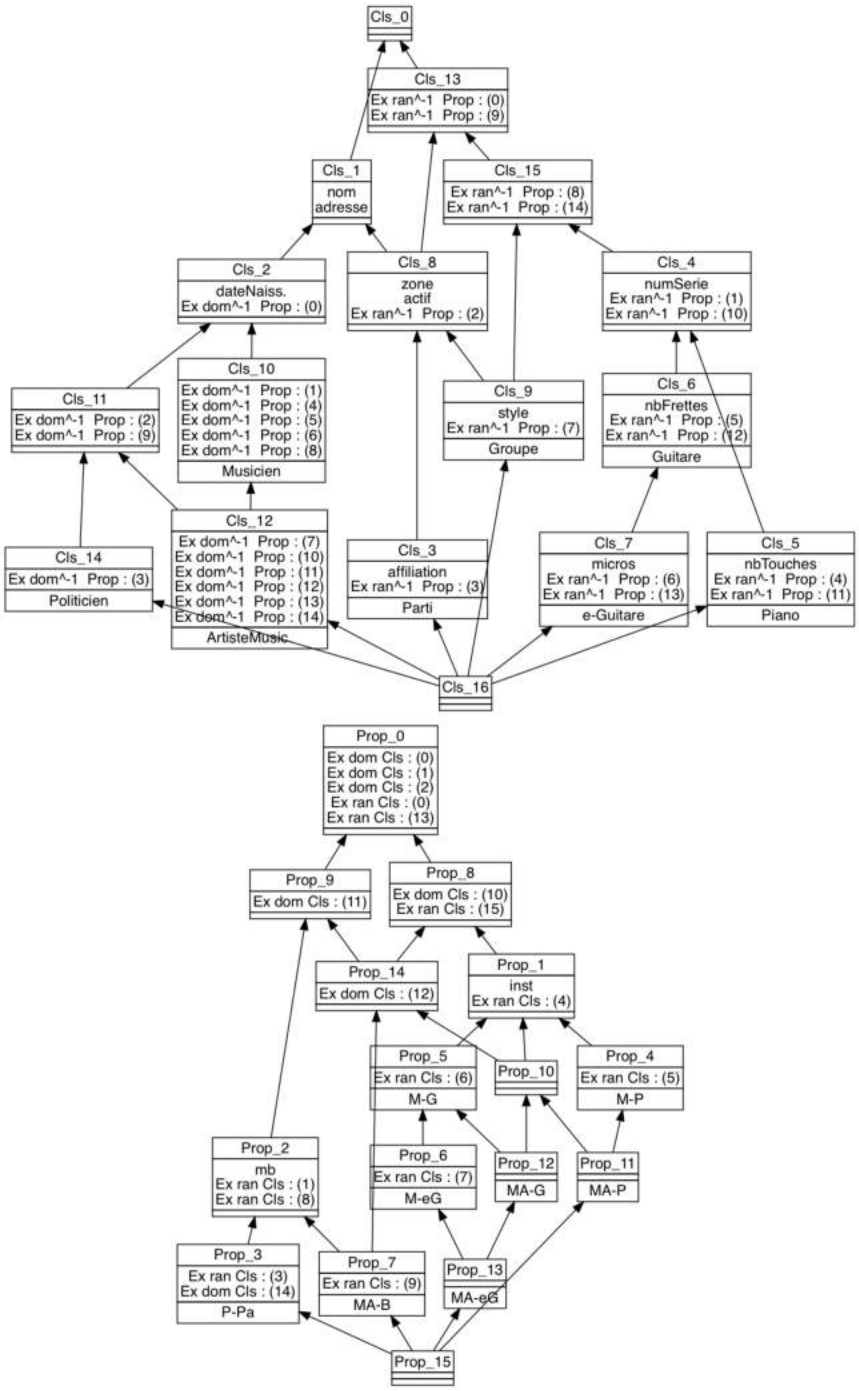


Figure 7.2: Treillis au point fixe de la RCF du tableau 7.1. Cls représente les concepts du treillis des classes, Prop (pour *property*) dénote les concepts des associations.

Proposition 12. *Considérons un concept $C_A = (X, Y)$ du treillis des associations formé par une telle famille relationnelle. Soit $Dom_Y = \{C \mid (\exists dom : C) \in Y\}$. On a* $\inf Dom_Y \in Dom_Y$

Démonstration. Soit un concept $C_A = (X, Y)$ du treillis des associations et soit $Dom_Y = \{C \mid (\exists dom : C) \in Y\}$ l'ensemble des attributs relationnels de Y. Pour chaque association $u \in X$, par construction, il existe un unique domaine de définition : la classe d. Soit D_X l'ensemble des domaines de définitions des

Classes	nom	dateNais.	adresse	affiliation	numSerie	nbTouches	nbFrettes	micros	zone	actif	style
Politicien	×	×	×								
Musicien	×	×	×								
ArtisteMusic.	×	×	×								
Piano					×	×					
Guit.					×		×				
e-Guit.					×		×	×			
Groupe	×		×						×	×	×
Parti	×		×	×					×	×	

association	instrument	membreDe
M-G	×	
AM-G	×	
M-eG	×	
AM-eG	×	
M-P	×	
AM-P	×	
AM-B		×
P-Pa		×

	Relation Domaine (dom)								Relation Codomaine (ran)							
	Politicien	Musicien	ArtisteMusic.	Piano	Guitare	e-Guitare	Groupe	Parti	Politicien	Musicien	ArtisteMusic.	Piano	Guit.	e-Guit.	Groupe	Parti
M-G		×											×			
AM-G			×										×			
M-eG		×												×		
AM-eG			×											×		
M-P		×										×				
AM-P			×									×				
AM-B			×												×	
P-Pa	×															×

Tableau 7.1: RCF correspondant à la figure 7.1. Les associations sont représentées par des paires $V - W$, où V est l'initiale de l'ensemble domaine, et W du codomaine de l'association. Par exemple M-G représente l'association entre Musicien et Guitare. *e-Guit.* est l'abbréviation pour la classe *Guitare électrique.*

éléments de X, c'est-à-dire $D_X = \{d \mid u \in X \wedge u$ défini sur $d\}$. Sur le treillis des classes, $C_D = (D''_X, D'_X)$ est le plus petit concept formel contenant D_X. Puisque pour tout $u \in X$, le domaine de u est dans D_X, alors $(\exists dom : C_D) \in Y$. Supposons un second attribut $(\exists dom : C) \in Y$. On en déduit alors que C contient tous les domaines des éléments de X, à savoir D_X. Donc $C_D \subseteq C$. Ainsi, puisque pour tout $C \in Dom_Y$ on a $C_D \subseteq C$, et que $C_D \in Dom_Y$, on a $C_D = \inf Dom_Y$ (et $C_D \in Dom_Y$).

□

Exemple 36. *Considérons sur la figure 7.2 le concept Prop 9. Les attributs relationnels «domaine» englobent les concepts du treillis des classe 0, 1, 2, 11. Les associations de l'extension sont P-Pa, MA-B, MA-P, MA-G, MA-eG. Les domaines décrits sont P (*Politician*) et MA (*Musical Artist*). Le plus petit concept contenant P et MA est* $(\{P, MA\}'', \{P, MA\}')$*, c'est-à-dire le concept 11 des classes. Puisque tout concept référé par un attribut de Prop 9 doit au moins contenir P et MA, on vérifie que* $Cls11 = \inf\{Cls0, Cls1, Cls2, Cls11\}$.

La proposition 12 peut être énoncée de manière analogue pour les codomaines. Finalement, la compression relationnelle limite chaque concept du treillis des associations à un unique domaine et un unique codomaine. Il est à noter que les attributs relationnels du contexte des classes présentent des liens vers les concepts des treillis des associations. De fait, ils ont un rôle dans la construction des classes, mais l'information de ces attributs est décrite «en double» dans le treillis des associations.

On utilise maintenant l'ARC sur la même famille relationnelle de concepts. Mais, grâce à la compression relationnelle sur les treillis des associations, l'ambiguïté due aux multiples possibilités de domaine et codomaine est levée. Ainsi, dans chaque concept on trouve la description d'une unique association pour le schéma final (cf.

proposition 12). La figure 7.3 présente le treillis des associations, au point fixe, avec compression relationnelle.

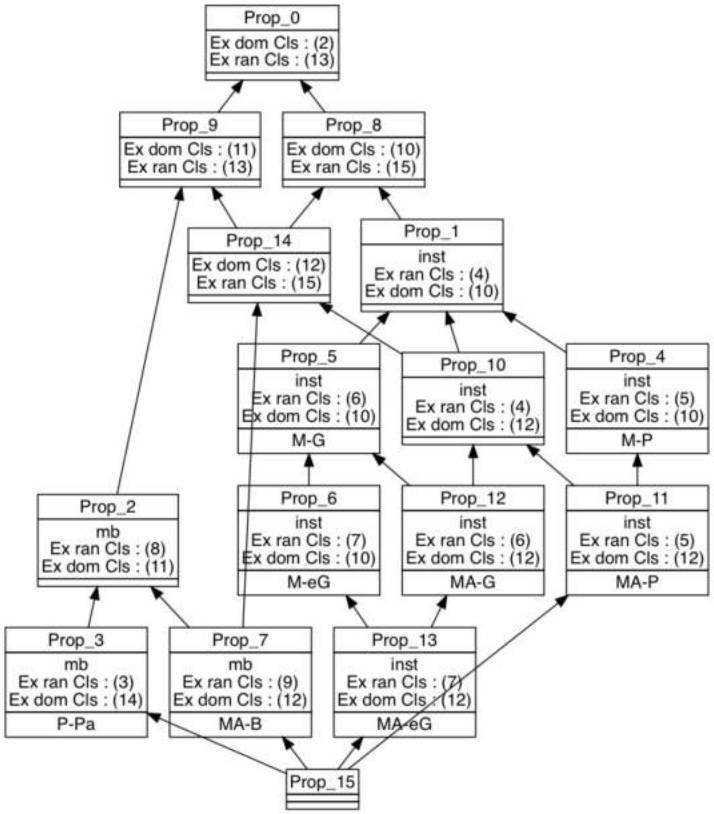


Figure 7.3: Treillis des associations, au point fixe. Les attributs d'un noeud représentent complètement l'intension du concept décrit : les intensions ne sont pas réduite pas la compression «non relationnelle» appliquée aux diagrammes de Hasse.

On peut donc maintenant, enrichir le diagramme de classe originel. La procédure manuelle proposée dans (Rouane-Hacene *et al.*, 2011) est la suivante :

— Conserver les classes originales.

— Dans une approche récursive partant de la classe *bottom* : étudier tous les super concepts, et intégrer au diagramme de classes les concepts pertinents, réitérer jusqu'au haut du treillis. Conserver les liens d'héritage.

— Nommer les entités.

Bien que cette approche amène un contrôle absolu sur le schéma à concevoir, l'évaluation manuelle est fastidieuse. Nous proposons donc la procédure automatisée suivante :

— Tronquer la partie supérieure du treillis de classes : tous les concepts n'ayant que des attributs relationnels dans leur intension Dans notre exemple, on retire donc les concepts 0, 13 et 15.
— Retirer tous les attributs relationnels des concepts des classes conservés puisque le treillis des associations couvre cette information.
— Pour chaque concept, créer la classe correspondante dans l'ontologie qui porte l'intension (non-relationnelle) du concept.
— Créer les liens de sous-classe suivant la hiérarchie du treillis.
— Tronquer la partie supérieure du treillis de associations : tous les concepts dont au moins l'un des deux attributs relationnels réfère à un concept supprimé lors de la première étape.
— Pour chaque concept, créer l'association dont le domaine et codomaine sont repérés par les attributs relationnels correspondants de l'intension.
— Créer les liens de sous-association suivant la hiérarchie des concepts.

En respectant cette procédure on obtient le schéma en figure 7.4. Premièrement, on constate que toutes les classes et associations du schéma 7.1 sont conservées. Ensuite, on peut noter, d'une part, que de nouvelles classes ont émergé, ainsi que des liens d'héritage entre les classes (représentés par des flèches au trait plein) ; et, d'autre part, que l'information représentée par les attributs de classe a été factorisée. De plus, de nouvelles associations ont émergé pour caractériser les correspondances des nouvelles classes. Enfin, un héritage est aussi apparu entre les associations (représenté par les flèches en pointillés). On constate donc un enrichissement du schéma ainsi qu'une restructuration due à la factorisation.

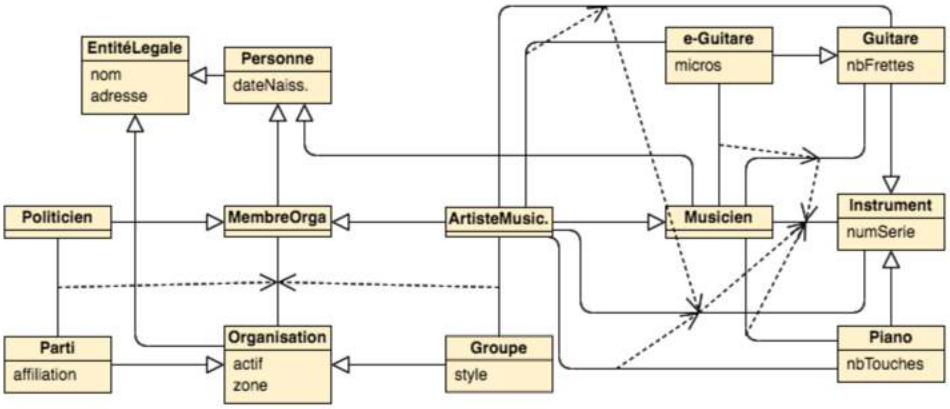


Figure 7.4: Schéma de classes après restructuration

Ainsi, la compression relationnelle vient compléter l'état de l'art pour la conception d'un schéma de manière automatique en proposant la découverte des associations reliant les classes nouvellement découvertes.

7.3 Détection des anomalies dans le typage des données

Dans cette seconde section, nous indiquons comment exploiter les règles d'association explicites pour détecter les anomalies de plongement d'un ensemble de données vers un modèle (typage).

Ces anomalies de typage peuvent se traduire de plusieurs manières, notamment par l'oubli ou l'ajout fortuit de certaines propriétés pour un objet. Voyons ici un exemple inspiré de *DBpedia*.

En utilisant les points d'accès ouverts de *DBpedia*, nous avons pu considérer l'existence des classes que l'on présente à la figure 7.5 : les classes `Person`, `Organisation` et `Instrument` (de musique), ainsi que certaines de leurs sous-classes.

Considérons à présent l'ensemble des personnes du contexte K_{Pers} de 7.2, extrait de *DBpedia*. Par une requête *SPARQL* sur le point d'accès ouvert de *DBpedia*, on

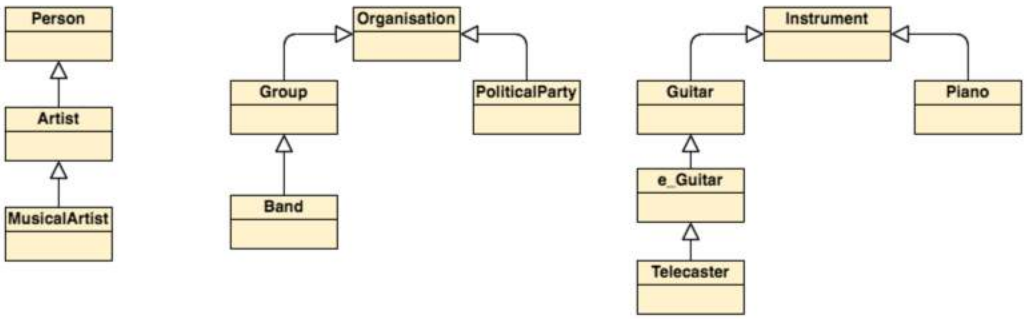


Figure 7.5: Diagramme de classe extrait de *DBpedia*.

peut extraire l'ensemble des groupes de musique dont les artistes sont membres ou ont été membres. Nous n'avons, pour l'exemple, conservé que ceux qui nous semblaient les plus emblématiques de leur carrière. On extrait également le parti politique de chacun des présidents des États-Unis considérés. Les groupes de musique et les partis politiques étant des instances de la classe *Organisation* de *DBpedia*, ils sont répertoriés dans le contexte K_{Org}. En ce qui concerne les instruments de musique, seules sont disponibles des ressources génériques. Nous avons extrapolé en créant des instances particulières pour mieux correspondre à la réalité.

Exemple 37. *En effet, George Harrison et James Hetfield jouent ou ont joué de la guitare sur un modèle de Fender Télécaster. Dans* DBpedia *on trouve les triplets <George Harrison, a pour instrument, Fender Télécaster> et <James Hetfield, a pour instrument, Fender Télécaster>. Ici «Fender Télécaster» est une ressource générique, c'est-à-dire qu'elle décrit* une *«Fender Télécaster». Toutefois, en réalité, les guitares considérées sont physiquement deux objets différents. Donc on crée les ressources spécifiques* la *«Fender Télécaster de George Harrison» et* la *«Fender Télécaster de James Hetfield»*

Pour plus de lisibilité, plutôt que de les appeler $instrument_1$, ..., $instrument_n$, on leur affecte un nom de la forme $V - W$ où V est l'ensemble d'initiales de l'artiste et W est parmi T (Fender Telecaster), Ibz (Guitare electrique Ibanez), G (Guitare),

ESP (Guitare Electrique ESP), F (Fender Bass), S (Saxophone), D (Drum). Ils sont répertoriés dans le contexte K_{Instru}.

K_{Pers}	MusicalArtist	Artist	Person
Jennifer Batten	×	×	×
George Harrison	×	×	×
John Lennon	×	×	×
Paul McCartney	×	×	×
Ringo Starr	×	×	×
James Hetfield	×	×	×
Kirk Hammet	×	×	×
Cliff Burton	×	×	×
AJ Roach			×
Jimmy Carter			×
George Bush			×
Bill Clinton			×

K_{Org}	Band	Group	Political Party	Organisation
Beatles	×	×		×
Metallica	×	×		×
Back	ok ×	×		×
Todd Sickafoose	×	×		×
Republican			×	×
Democratic			×	×

K_{Instru}	Telecaster	e_Guitar	Guitar	Piano	Instrument
GH-T	×	×	×		×
JH-T	×	×	×		×
JB-Ibz		×	×		×
JH-ESP		×	×		×
KH-ESP		×	×		×
AJR-G			×		×
JL-G			×		×
CB-F					×
BC-S					×
JL-P				×	×
PMC-G			×		×
RS-D					×

Tableau 7.2: Ensemble des contextes formels que l'on peut générer à partir des données

Les relations d'appartenance à une organisation (isMemberOf) ainsi que l'instrument joué (instrument) sont également connues. Elles sont répertoriées dans les relations formelles des tables 7.3.

Considérons une personne ; chaque instrument joué par celle-ci et chaque organisation dont elle est membre contribuent à sa description. Ainsi, on utilise l'opérateur $\exists$ pour les relations *isMemberOf* et *instrument*. Un instrument peut aussi être décrit par l'existence d'un de ses utilisateurs, on utilise alors l'opérateur $\exists$ pour la relation inverse $instrument^{-}$. Par contre la description d'une organisation n'est pertinente que par les points communs de tous ses membres. Ainsi, nous avons appliqué l'opérateur $\forall\exists$ sur la relation $isMemberOf^{-1}$.

Après une itération par ARC, on trouve les treillis illustrés en figure 7.6.

On peut observer les règles suivantes, comme étant les deux règles, d'association

	$R_{isMemberOf}$						$R_{instrument}$											
	Beatles	Metallica	Beck	Todd Sickafoose	Republican	Democratic	GH-T	JH-T	JB-Ibz	JH-ESP	KH-ESP	AJR-G	JL-G	CB-F	BC-S	JL-P	PMC-G	RS-D
Jennifer Batten			×						×									
George Harrison	×						×											
John Lennon	×												×			×		
Paul McCartney	×																×	
Ringo Starr	×																	×
James Hetfield		×						×		×								
Kirk Hammet		×									×							
Cliff Burton		×												×				
AJ Roach				×								×						
Jimmy Carter						×												
George Bush					×													
Bill Clinton						×									×			

Tableau 7.3: Table des relations

non exactes, de plus grande confiance extraites sur les treillis `Organisation` et `Person` :

$$Org_1 \ : \ Band \rightarrow \forall\exists isMemberOf^{-1} : (MusicalArtist) \ [\text{confiance} : 75\%]$$

$$Pers_8 \ : \ \exists isMemberOf : (\ Band\ ) \rightarrow MusicalArtist, Artist \ [\text{confiance} : 89\%]$$

La règle du concept Org_1 stipule qu'un groupe de musique (*Band*) est composé uniquement d'artistes musiciens. Dans la même direction, la seconde règle que si une personne est membre d'au moins un groupe de musique elle est artiste musicien. Bien que ces règles semblent correspondre à des définitions, les confiances affichées ne sont pas de 100%. Est-il envisageable qu'un membre d'un groupe ne soit pas répertorié MusicalArtist ? Il semble plus probable que l'étiquette MusicalArtist ait été oubliée. Dans le présent cas, seule une instance, AJ Roach, est répertoriée comme faisant partie d'un groupe de musique (Band) mais n'est pas

un musicien professionnel (Musical Artist). Sur la page Wikipédia, on peut pourtant y lire qu'AJ Roach est effectivement un professionnel de la musique. Comme *DBpedia* vise à extraire, en données liées, toute l'information de Wikipédia, on peut effectivement considérer que le label MusicalArtist est passé entre les mailles du filet et aurait dû être présent sur *DBpedia*.

La confiance d'une règle d'association traduit, rappelons-le, le nombre d'objets possédant les attributs de la conclusion et de la prémisse par rapport à ceux n'ayant que la prémisse. Ainsi, le complémentaire de la confiance reflète le ratio du nombre d'objets ayant les attributs de la prémisse mais n'ayant pas ceux de la conclusion.

Au final, plus le jeu de données est important, plus la découverte d'anomalies ponctuelles est aisée grâce aux règles d'association. En pratique, on se munit d'un seuil de confiance au dessus duquel on considère qu'une règle d'association devrait être une implication, puis on procède comme suit :

— Ordonner les règles d'association par confiance
— Filtrer les règles d'association en dessous du seuil
— Pour toute règle conservée, étudier les instances validant la prémisse mais pas la conclusion.

La lisibilité que donne l'explicitation des règles permet de déterminer immédiatement si les instances ne vérifiant pas une règle sont à réévaluer.

7.4 Raffinement du schéma

Raffiner un schéma consiste à utiliser un jeu de données pour découvrir de nouvelles associations et de nouvelles classes dans le schéma. En ce sens, l'extraction de schéma est un cas particulier du raffinement, dans lequel le schéma est une classe

unique identifiant toutes les instances. Mehri et al. proposent d'utiliser l'AFC pour extraire un schéma depuis les données (Mehri et Valtchev, 2017). Toutefois une telle méhode n'inclut pas les associations. Dans une optique complémentaire, Hacene et al proposent d'utiliser l'ARC pour concevoir le schéma qui reflète au mieux les données (Hacene *et al.*, 2008). Dans cet article, l'hypothèse est toutefois restrictive : le type des entités doit être connu. Même si ce type est donné de manière générique, un traitement préalable, manuel ou non, est nécessaire. Nous présentons donc dans cette section un cas d'application dans lequel aucun schéma n'est disponible (par exemple c'est le cas des données liées ouvertes du web). Toutefois les données, avec leurs attributs et leurs relations, le sont. Dans une telle configuration, les treillis conçus via l'ARC permettent de proposer une version préliminaire d'un schéma. Celui-ci doit être raffiné pour séparer les classes qui sont l'expression simple de la distribution des données de celles qui représentent une description réelle du domaine de connaissances à formaliser. Nous nous attelons à montrer que la combinaison de la compression relationnelle et l'expression des attributs relationnels avec générateur canonique permet une interprétabilité supérieure à la version originale.

Munissons-nous de l'ensemble de données de la section précédente : des personnes, des instruments de musique et des organisations. Dans cette expérience, nous ne spécifions toutefois à aucun moment à la machine les classes de ces objets.

Le cadre que nous proposons est le suivant : un contexte unique contient tous les objets. Ce contexte, dans sa forme minimale, est diagonal : chaque objet crée un attribut qui le représente, par exemple son nom ou son URI. Soit K, le contexte formel diagonal 30×30 tel que l'ensemble des objets soit la réunion des ensembles d'objets de K_{Pers}, K_{Org} et K_{Instru}. Nous ne reproduisons pas cette table trop volumineuse. De plus, s'ils avaient existé, les attributs des objets seraient ajoutés comme des colonnes supplémentaires. (Un exemple peut être consulté dans le

tableau 7.4.)

	Beatles	*Beck*	*Democratic*	*Grammy*	*centenaire*
Beatles	×			×	
Beck		×			
Democratic			×		×

Tableau 7.4: Contexte diagonalisé pour les organisations *Beatles*, *Beck*, *Democratic* ayant les attributs *Grammy* précisant si une organisation a obtenue le prix de musique *Grammy award* de l'album de l'année et *centenaire* précise si une organisation a plus de 100 ans

En plus de ce contexte, chaque type de relation lie ce contexte avec lui-même. Dans notre exemple, nous utilisons les relations *isMemberOf* (abbrégée *mb*) et *instrument* (*inst*) détaillées au tableau 7.3. Les relations binaires sont bien entendu adaptées pour être au format 30×30, mais la relation d'incidence reste inchangée.

Aucun a priori ne peut être fait sur les classes qui vont être produites par ce processus. Ainsi, les relations sont utilisées avec leur inverse et les opérateurs de graduation $\exists$ et $\forall\exists$ sont utilisés conjointement.

Après avoir atteint le point fixe, le treillis produit est celui que l'on peut trouver en figure 7.7. En raison du manque d'espace, nous n'avons affiché que l'ensemble des concepts avec au moins deux objets, et nous avons indiqué l'extension des concepts minimaux conservés pour l'affichage. Pour les attributs relationnels référant à un concept supprimé, on affiche dans la référence l'extension du concept (qui est donc est de taille 0 ou 1). Voyons maintenant les suggestions, en terme de schéma, qu'un tel treillis peut fournir.

Tout d'abord, on peut noter les concepts 33, 36 et 40 qui contiennent respectivement dans leur extension, tous les instruments, toutes les personnes et toutes les organisations. Ainsi, il est possible par le typage inhérent aux relations, de découvrir les classes racines. Puisque ces classes ont émergé, le reste du traitement

peut se poursuivre comme dans (Hacene *et al.*, 2008). En appliquant les règles de la section 5 de (Rouane *et al.*, 2007), on peut extraire des concepts trouvés, un ensemble de restrictions qui correspondent à l'intension de ce concept. Ensemble, elles forment une base de connaissance. Toutefois, les définitions cycliques sont communes. Par exemple, on trouve la paire de concepts 36-40, où $\exists mb^{-1} : (36)$ est dans l'intension du concept 40 et $\exists mb : (40)$ est dans l'intension du concept 36. Ce n'est pas un cas isolé puisqu'on trouve aussi des définitions cycliques dans les paires 32-33, 38-42, 44-46, 46-47, etc.

En utilisant maintenant l'ARC pourvue de sa compression relationnelle et de ses attributs relationnels désambiguïsés, on obtient la figure 7.8. On peut y voir que l'intension du concept 40, originellement $\{\exists mb^{-1} : (0), \forall\exists mb^{-1} : (0), \exists mb^{-1} : (36), \forall\exists mb^{-1} : (36)\}$, devient $\{\forall\exists mb^{-1} : (\top)\}$, c'est-à-dire que le concept 40 est composé des instances qui «contiennent des membres». La figure 7.8 montre que les optimisations permettent la lecture directe des intensions.

Les attributs relationnels sont donc maintenant directement exploitables. Par exemple, l'attribut du concept 33, $\forall\exists inst^{-}.\top$, représente la classe codomaine de la relation *instrument*. Seuls des instruments peuvent être joués, donc on peut définir le concept 33 comme la classe des "Instruments".

En définissant récursivement tous les attributs relationnels, comme dans le tableau 7.5 où $\triangleq$ signifie la définition, on peut concevoir un schéma de classes. Pour cela, on définit récursivement tous les attributs relationnels. Lorsqu'un attribut relationnel n'est pas considéré pertinent, on le supprime avec son concept-attribut (par exemple le concept 43). On considère les classes ainsi formées. Puis, pour chaque concept C_1 ayant un attribut relationnel $\rho r : C_2$ on introduit l'association r entre C_1 et C_2. Finalement, on obtient le schéma présent en figure 7.9.

#	définition	#	définition
33	$Instrument \triangleq \forall\exists inst^{-}.\top$	40	$Organisation \triangleq \forall\exists mb^{-}.\top$
36	$Personne \triangleq \forall\exists mb.\top$	32	$Musicien \triangleq \forall\exists inst^{-}.\top$
44	$Groupe \triangleq \forall\exists mb^{-}.Musicien$	46	$MusicienPro \triangleq \forall\exists mb.Groupe$
...	...	...	...

Tableau 7.5: Extrait du tableau de définition récursive des attributs relationnels par concept

7.5 Conclusion

Dans ce chapitre, nous avons exposé des pistes de développement de l'ingénierie de connaissance en utilisant l'ARC dans sa version aux attributs relationnels explicites. Grâce à sa capacité à considérer les attributs des objets ainsi que les liens les mettant en relation, l'ARC permet de détecter les anomalies de typage dans un jeu de données et à générer des treillis de concepts. Les concepts représentent des descriptions des classes possibles pour la constitution d'un schéma de données. Le treillis traduit la relation d'héritage entre les classes et les attributs relationnels permettent de modéliser les associations entre les classes. Dans des travaux futurs, il conviendrait de montrer que les trois applications présentées dans ce chapitre peuvent être utilisées sur des jeux de données réels.

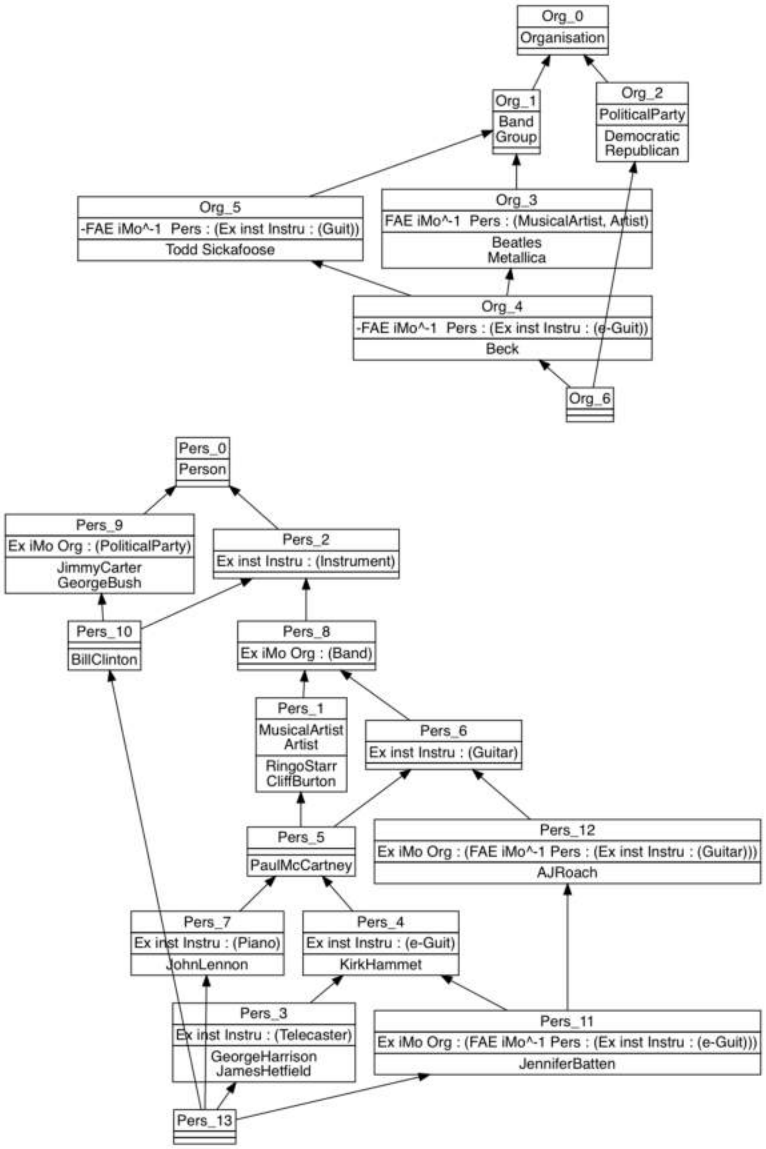


Figure 7.6: Treillis au point fixe des contextes K_{Pers} et K_{Org} en utilisant la compression relationnelle et l'explicitation des attributs relationnels avec les générateurs canoniques.

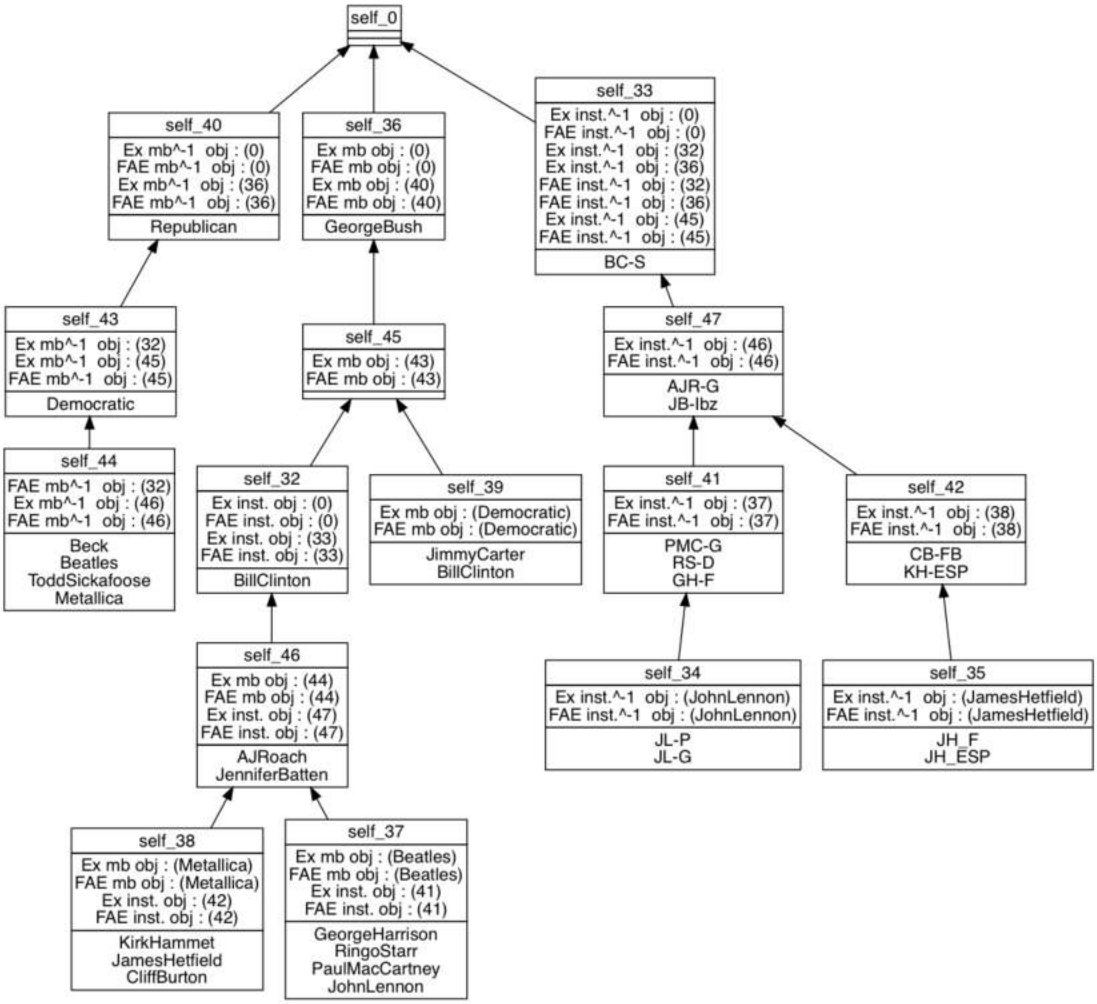


Figure 7.7: Treillis au point fixe du contexte K obtenu avec l'ARC

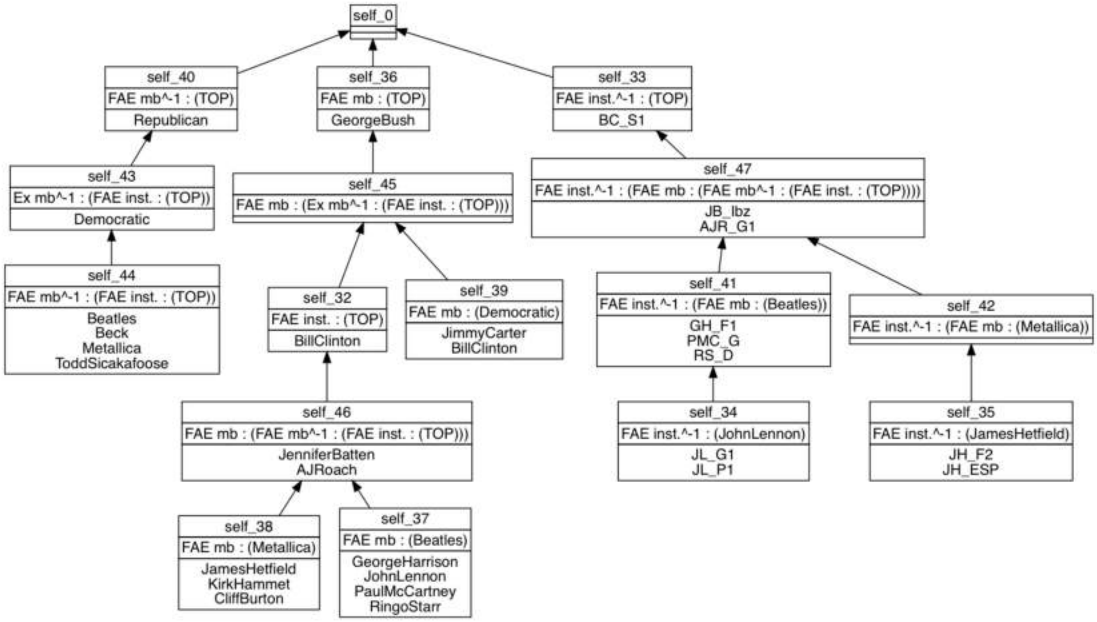


Figure 7.8: Treillis au point fixe du contexte K après optimisations

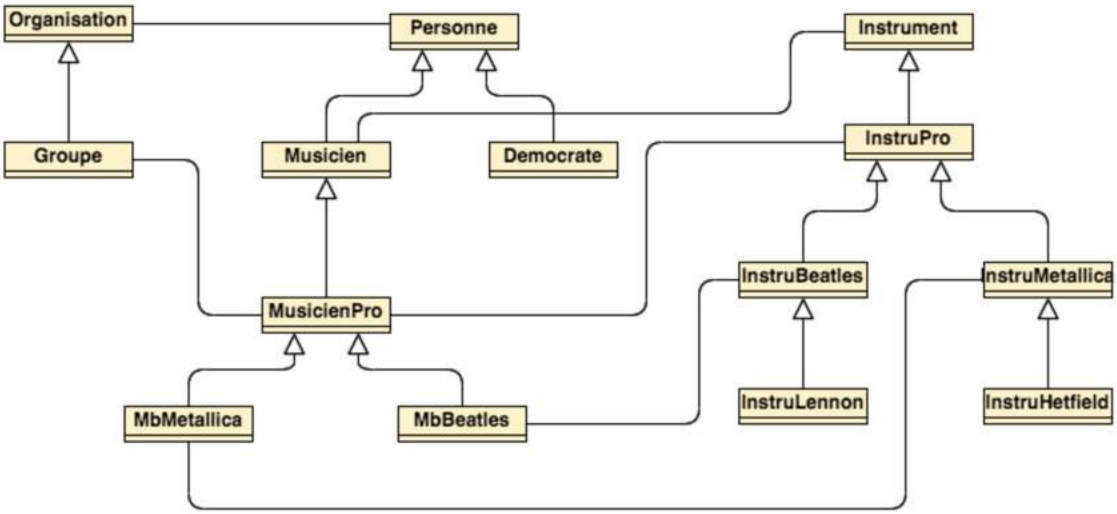


Figure 7.9: Schéma raffiné du contexte K

CONCLUSION

La discipline d'*extraction de connaissance* a été développée pour répondre aux besoins de synthèse et d'analyse des jeux de données trop larges pour être appréhendés par l'humain. Les méthodes de ce domaine ont pour but de mettre en lumière la «réalité du terrain» exprimée au travers d'un jeu de données. Au chapitre 1, on commence par s'interroger notamment sur la définition d'*extraire des connaissances* d'un jeu de données. Bien que la sémantique ne soit pas universelle, nous avons adopté la définition de (Fayyad *et al.*, 1996), un des articles les plus cités de ce domaine. Dans ce dernier, les données sont considérées comme des entrées, et l'information comme l'ensemble des agrégations possibles de celles-ci. L'extraction de connaissance est alors la discipline de conception d'un modèle de connaissance basé sur les informations.

Plusieurs formes de sortie sont donc possibles. Certains processus visent à extraire directement un modèle de connaissances, par exemple sous la forme d'une ontologie, alors que d'autres visent à présenter à l'utilisateur les informations les plus intéressantes (la mesure d'intérêt étant laissée au soin de ce dernier). Cette seconde alternative utilise la fouille de données pour extraire l'information d'un jeu de données. Toutefois, les jeux de données étant relationnels, c'est-à-dire présentant différents type d'objets et des liens entre eux, et la fouille de données traitant des ensembles de données non relationnelles, deux choix sont possibles. Soit on peut artificiellement homogénéiser le jeu de données pour n'avoir qu'un seul type (complexe) de données, soit on développe des méthodes de la fouille de données multirelationnelles qui exploite les liens entre les objets. C'est évidemment un compromis entre complexité algorithmique et exhaustivité des résultats.

De plus, l'information extraite doit alors facilement être interprétable par l'utilisateur. Parmi les différentes formes possibles, cette thèse se concentre sur les règles d'associations, une forme d'information présentant les co-occurences entre caractéristiques des objets. Ces règles d'association sont de la forme *si un objet a les caractéristiques «a» et «b» alors il a aussi «c», «d» et «e»*. Les mesures d'intérêt les plus classiques pour ces règles sont le *support*, c'est-à-dire le nombre d'objets concernés par une règle, et la *confiance*, c'est-à-dire la véracité d'une règle.

L'ensemble de toutes les règles qui peuvent être produites à partir d'un jeu de données est exponentiel par rapport au nombre de caractéristiques existantes. De plus cet ensemble présente une forte redondance. Il convient donc de chercher une base de représentation de cet ensemble. On considère une base de règles comme étant représentative si, par un mécanisme d'inférence, on peut générer exhaustivement et exclusivement depuis celle-ci toutes les règles du jeu de données. De surcroît, il faut que les règles de la base, ainsi que celles déduites, puissent être évaluées en fonction de mesures d'intérêt, ici la confiance et le support. Enfin, on cherche évidemment la base de représentation la plus petite possible. Le chapitre 2 présente l'analyse formelle de concept (AFC), une méthode mathématique détectant, dans un jeu de données binarisé, les groupes d'objets partageant des caractéristiques communes, ces groupes étant appelés concepts formels. En organisant l'ensemble des concepts formels dans un treillis, l'AFC permet d'extraire de cette hiérarchie une base de représentation compacte répondant aux critères susmentionnés. Toutefois, l'AFC est une méthode limitée aux jeux de données non relationnels.

Le chapitre 3 présente différentes alternatives de la littérature qui proposent un élargissement de l'AFC au paradigme multi-relationnel. Parmi celles-ci, nous nous sommes intéressés plus particulièrement à l'analyse relationnelle de concepts (ARC) qui a été éprouvée dans plusieurs domaines et qui s'aligne avec les forma-

lismes des ontologies. L'ARC utilise des mécanismes de propositionnalisation pour intégrer l'information relationnelle dans la description des objets qui, par la suite, sont soumis à une méthode d'AFC. Le processus est itératif et s'arrête lorsqu'un point fixe est atteint. Toutefois, les liens entre les objets peuvent induire des cycles, par exemple lorsque les relations sont bi-directionnelles. Ces circuits, incorporés à l'information relationnelle, complexifient le processus d'interprétation. Ainsi, dans la littérature deux types d'approches sont mises en avant : l'utilisation de l'ARC lorsque les données ne présentent aucun circuits ou l'interprétation des circuits, ce dernier empêchant la production de règles indépendantes.

Cette thèse vient, au travers du chapitre 4, contribuer au développement de l'ARC pour permettre l'extraction de règles d'association. On montre notamment comment contourner la présence de circuits dans les descriptions des objets et ce, sur un jeu de données multirelationnel binarisé quelconque. De plus, on y renforce la théorie de l'ARC notamment en redéfinissant la notion de concept formel, en démontrant mathématiquement l'intérêt de l'ARC par rapport à l'AFC, et en proposant certaines améliorations visant à supprimer la redondance dans les règles qui pourrait apparaître du fait de la relation inhérentes entre les caractéristiques relationnelles.

Finalement, la seconde partie de la thèse est un ensemble d'applications qui montrent l'utilité de ces améliorations, en particulier les règles d'associations pouvant être utilisées de manière indépendante de l'ensemble des treillis de concepts produits. Le chapitre 5 présente des règles liant des caractéristiques de pièces manufacturées par moulage d'aluminium à des problèmes de production. Elles ont permis de découvrir l'origine des problèmes qui n'avaient alors pas été envisagés, tel que l'impact de la dilatation du piston acheminant l'aluminium en fusion dans le moule de presse sur la qualité du produit. Le chapitre 6 propose une application des règles en psycholinguistique. Il présente des dictionnaires produits

dans le cadre d'un jeu sérieux, dont le but est de définir le plus petit nombre de mots nécessaire pour que tous les mots employés dans une définition soient eux même définis. Les règles d'associations ont, dans ce contexte, permis de mettre en relation les caractéristiques structurelles de la représentation en graphe de ces dictionnaires et les descriptions psycholinguistiques des mots utilisés. Le dernier chapitre de cette seconde partie, le chapitre 7, introduit l'ARC et les treillis sans attribut circulaire comme outil polyvalent de l'ingénierie de la connaissance agissant à différents niveaux. On y trouve l'ARC comme méthode pour restructurer un schéma de données, détecter de nouvelles classes et relations, implicites dans les données, et de détecter les erreurs de typage.

Ce dernier cas mérite une étude plus approfondie, notamment avec un jeu de données réel. De plus, dans tous les cas applicatifs présentés, on a pu montrer que l'ARC permettait de mettre en évidence des observations pertinentes. Toutefois, il reste encore de nombreuses pistes pour tendre vers la complétion de la méthode et la solidification de sa théorie. Notamment il serait pertinent, en plus de savoir que l'algorithme converge, d'être capable de déterminer les facteurs impactant le temps de convergence. Puis, dans une perspective plus lointaine, il serait intéressant de proposer une version de l'ARC permettant d'intégrer des valeurs numériques et catégorielles et de gérer les relations ayant elles-même une valuation. Enfin, la description en formules logiques des intensions des concepts, qui sont aujourd'hui limitées à l'opérateur de conjonction, pourrait amener une richesse supplémentaire à l'expressivité des règles d'association.

RÉFÉRENCES

Agrawal, R. *et al.* (1993). Mining association rules between sets of items in large databases. Dans *Acm sigmod record*, volume 22, 207–216. ACM.

Agrawal, R. *et al.* (1996). Fast discovery of association rules. *Advances in knowledge discovery and data mining*, *12*(1), 307–328.

Alam, M., Chekol, M. W., Coulet, A., Napoli, A. et Smaïl-Tabbone, M. (2013). Lattice based data access (lbda) : An approach for organizing and accessing linked open data in biology.

Alavi, M. et Leidner, D. E. (2001). Knowledge management and knowledge management systems : Conceptual foundations and research issues. *MIS quarterly*, 107–136.

Andréka, H., Németi, I. et van Benthem, J. (1998). Modal languages and bounded fragments of predicate logic. *Journal of philosophical logic*, *27*(3), 217–274.

Andrews, S. (2009). In-close, a fast algorithm for computing formal concepts.

Armstrong, W. W., Nakamura, Y. et Rudnicki, P. (2002). Armstrong's axioms. *Journal of formalized Mathematics*, *14*.

Arp, R., Smith, B. et Spear, A. D. (2015). *Building ontologies with basic formal ontology*. Mit Press.

Atkinson, R. (1999). Project management : cost, time and quality, two best guesses and a phenomenon, its time to accept other success criteria. *International journal of project management*, *17*(6), 337–342.

Azmeh, Z., Driss, M., Hamoui, F., Huchard, M., Moha, N. et Tibermacine, C. (2011). Selection of composable web services driven by user requirements. Dans *2011 IEEE International Conference on Web Services*, 395–402. IEEE.

Azmeh, Z., Mirbel, I. et Crescenzo, P. (2013). Highlighting stakeholder communities to support requirements decision-making. In *International Working Conference on Requirements Engineering : Foundation for Software Quality* 190–205.

Baader, F., Calvanese, D., McGuinness, D., Patel-Schneider, P., Nardi, D. *et al.* (2003). *The description logic handbook : Theory, implementation and applications*. Cambridge university press.

Baixeries, J., Szathmary, L., Valtchev, P. et Godin, R. (2009). Yet a faster algorithm for building the hasse diagram of a concept lattice. Dans *International Conference on Formal Concept Analysis*, 162–177. Springer.

Bendaoud, R., Hacene, A. M. R., Toussaint, Y., Delecroix, B. et Napoli, A. (2007). Text-based ontology construction using relational concept analysis.

Berardi, D., Calvanese, D. et De Giacomo, G. (2005). Reasoning on uml class diagrams. *Artificial intelligence*, *168*(1-2), 70–118.

Bina, B., Schulte, O., Crawford, B., Qian, Z. et Xiong, Y. (2013). Simple decision forests for multi-relational classification. *Decision Support Systems*, *54*(3), 1269–1279.

Blondin Massé, A., Chicoisne, G., Gargouri, Y., Harnad, S., Picard, O. et Marcotte, O. (2008). How is meaning grounded in dictionary definitions ? Dans *Proceedings of the 3rd Textgraphs Workshop on Graph-Based Algorithms for Natural Language Processing*, 17–24. Association for Computational Linguistics.

Brachman, R. J. (1979). On the epistemological status of semantic networks. In *Associative networks* 3–50. Elsevier.

Browne, C. (2013). The new general service list : A core vocabulary for EFL students & teachers.

Browne, C. (2014). A new general service list : The better mousetrap we've been looking for. *Vocabulary Learning and Instruction*, *3*(2), 1–10.

Browne, C. *et al.* (2013). The new general service list : Celebrating 60 years of vocabulary learning. *The Language Teacher*, *37*(4), 13–16.

Brysbaert, M. et New, B. (2009). Moving beyond kučera and francis : A critical evaluation of current word frequency norms and the introduction of a new and improved word frequency measure for american english. *Behavior research methods*, *41*(4), 977–990.

Brysbaert, M., Warriner, A. et Kuperman, V. (2014). Concreteness ratings for 40 thousand generally known english word lemmas. *Behavior research methods*, *46*(3), 904–911.

BSA (2016). What's the big deal with data ? http://download.microsoft.com/documents/en-us/sam/bsadatastudy_en.pdf. Accessed : 2020-04-17.

Bühmann, L., Lehmann, J. et Westphal, P. (2016). Dl-learner—a framework for inductive learning on the semantic web. *Journal of Web Semantics*, *39*, 15–24.

Cai, D., Shao, Z., He, X., Yan, X. et Han, J. (2005). Community mining from multi-relational networks. Dans *European Conference on Principles of Data Mining and Knowledge Discovery*, 445–452. Springer.

Carbonnel, J., Huchard, M. et Gutierrez, A. (2015). Variability representation in product lines using concept lattices : feasibility study with descriptions from wikipedia's product comparison matrices. *1434*.

Carbonnel, J., Huchard, M. et Nebut, C. (2018). Towards the extraction of variability information to assist variability modelling of complex product lines. Dans *Proceedings of the 12th International Workshop on Variability Modelling of Software-Intensive Systems*, 113–120. ACM.

Cerbah, F. (2008). Learning highly structured semantic repositories from relational databases. Dans *European Semantic Web Conference*, 777–781. Springer.

Chen, P. P.-S. (1976). The entity-relationship model—toward a unified view of data. *ACM transactions on database systems (TODS)*, *1*(1), 9–36.

Davey, B. A. et Priestley, H. A. (2002). *Introduction to lattices and order*. Cambridge university press.

David, J. (2007). Association rule ontology matching approach. *International Journal on Semantic Web and Information Systems (IJSWIS)*, *3*(2), 27–49.

David, J., Euzenat, J. et Vizzini, J. (2018). Linkky : Extraction de clés de liage par une adaptation de l'analyse relationnelle de concepts.

De Maio, C., Fenza, G., Gallo, M., Loia, V. et Senatore, S. (2014). Formal and relational concept analysis for fuzzy-based automatic semantic annotation. *Applied intelligence*, *40*(1), 154–177.

Dehaspe, L. et Toivonen, H. (1999). Discovery of frequent datalog patterns. *Data Mining and knowledge discovery*, *3*(1), 7–36.

Dolques, X., Le Ber, F., Huchard, M. et Grac, C. (2016). Performance-friendly rule extraction in large water data-sets with aoc posets and relational concept

analysis. *International Journal of General Systems*, *45*(2), 187–210.

Džeroski, S. (2003). Multi-relational data mining : an introduction. *ACM SIGKDD Explorations Newsletter*, *5*(1), 1–16.

El Hamdouni, A.-E., Seriai, A.-D. et Huchard, M. (2010). Component-based architecture recovery from object oriented systems via relational concept analysis. Dans *CLA : Concept Lattices and their Applications*, numéro 672, 259–270. University of Sevilla.

Encheva, S. (2015). For a better coordination between students learning styles and instructors teaching styles. *International Journal of Advanced Research in Artificial Intelligence*, *4*(3), 24–27.

Fayyad, U., Piatetsky-Shapiro, G. et Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI magazine*, *17*(3), 37–37.

Ferré, S. (2006). Negation, opposition, and possibility in logical concept analysis. In *Formal Concept Analysis* 130–145. Springer.

Ferré, S. (2015). A proposal for extending formal concept analysis to knowledge graphs. Dans *International Conference on Formal Concept Analysis*, 271–286. Springer.

Ferré, S. et Cellier, P. (2016). Graph-fca in practice. Dans *International Conference on Conceptual Structures*, 107–121. Springer.

Ferré, S. et Cellier, P. (2018). How hierarchies of concept graphs can facilitate the interpretation of rca lattices ?

Ferré, S. et Cellier, P. (2019). Graph-fca : An extension of formal concept analysis to knowledge graphs. *Discrete Applied Mathematics*.

Ferré, S. et Ridoux, O. (2000). A logical generalization of formal concept analysis. Dans *International Conference on Conceptual Structures*, 371–384. Springer.

Frank, R., Moser, F. et Ester, M. (2007). A method for multi-relational classification using single and multi-feature aggregation functions. Dans *European Conference on Principles of Data Mining and Knowledge Discovery*, 430–437. Springer.

Galárraga, L., Teflioudi, C., Hose, K. et Suchanek, F. M. (2015). Rule mining with amie+ fouille de règles avec amie.

Galárraga, L. A., Teflioudi, C., Hose, K. et Suchanek, F. (2013). Amie : association rule mining under incomplete evidence in ontological knowledge

bases. Dans *Proceedings of the 22nd international conference on World Wide Web*, 413–422. ACM.

Ganter, B. et Kuznetsov, S. O. (2001). Pattern structures and their projections. Dans *International conference on conceptual structures*, 129–142. Springer.

Ganter, B. et Wille, R. (1999). *Formal concept analysis : mathematical foundations*.

Garg, V. K. (2015). *Introduction to lattice theory with computer science applications*. Wiley Online Library.

Garrido, C. et Gutierrez, C. (2016). Dictionaries as networks : Identifying the graph structure of ogden's basic english. Dans *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics : Technical Papers*, 3565–3576.

G.E. (2012). Industrial internet : Pushing the boundaries of minds and machines. http://les.gereports.com/wp-content/uploads/2012/11/ge-industrial-internet-vision-paper.pdf. Accessed : 2020-04-17.

Geng, L. et Hamilton, H. J. (2006). Interestingness measures for data mining : A survey. *ACM Computing Surveys (CSUR)*, *38*(3), 9–es.

Goddard, C. (2018). *Minimal English for a global world*. Springer.

Goddard, C. et Wierzbicka, A. (1994). Introducing lexical primitives. *Semantic and Lexical Universals. Theory and Empirical Findings*, 31–54.

Godfrey-Smith, P. (2009). *Theory and reality : An introduction to the philosophy of science*. University of Chicago Press.

Gröner, G. et Staab, S. (2010). Categorization and recognition of ontology refactoring pattern.

Guédi, A. O., Huchard, M., Miralles, A. et Nebut, C. (2013). Practical application of relational concept analysis to class model factorization : lessons learned from a thematic information system. Dans *CLA : Concept Lattices and their Applications*, volume 1062, 9–20. CEUR WS.

Guesmi, S., Trabelsi, C. et Latiri, C. (2016a). Community detection in multi-relational bibliographic networks. Dans *International Conference on Database and Expert Systems Applications*, 11–18. Springer.

Guesmi, S., Trabelsi, C. et Latiri, C. (2016b). Comring : A framework for community detection based on multi-relational querying exploration. *Procedia*

Computer Science, *96*, 627–636.

Hacene, M. R., Napoli, A., Valtchev, P., Toussaint, Y. et Bendaoud, R. (2008). Ontology learning from text using relational concept analysis. Dans *2008 International MCETECH Conference on e-Technologies (mcetech 2008)*, 154–163. IEEE.

Han, J., Pei, J. et Yin, Y. (2000). Mining frequent patterns without candidate generation. *ACM sigmod record*, *29*(2), 1–12.

Harnad, S. (1990). The symbol grounding problem. *Physica D : Nonlinear Phenomena*, *42*(1-3), 335–346.

Hayes, P. J. (1981). The logic of frames. In *Readings in artificial intelligence* 451–458. Elsevier.

Hazman, M., El-Beltagy, S. R. et Rafea, A. (2011). A survey of ontology learning approaches. *International Journal of Computer Applications*, *22*(9), 36–43.

Hernán, M. et Robins, J. (2020). Causal inference : What if. *Boca Raton : Chapman & Hill/CRC*.

Horrocks, I., Kutz, O. et Sattler, U. (2006). The even more irresistible sroiq. *Kr*, *6*, 57–67.

Huchard, M. (2019). Formal concept analysis, a framework for knowledge structuring and exploration. applications to service directories and product lines.

Huchard, M., Roume, C. et Valtchev, P. (2002). When concepts point at other concepts : the case of uml diagram reconstruction. Dans *Proceedings of the 2nd Workshop on Advances in Formal Concept Analysis for Knowledge Discovery in Databases (FCAKDD)*, 32–43.

Józefowska, J. *et al.* (2008). On reducing redundancy in mining relational association rules from the semantic web. Dans *International Conference on Web Reasoning and Rule Systems*, 205–213. Springer.

Kapoor, P., Singh, P. K. et Cherukuri, A. K. (2020). Crime data set analysis using formal concept analysis (fca) : A survey. In *Advances in Data Sciences, Security and Applications* 15–31. Springer.

Kötters, J. (2016). Intension graphs as patterns over power context families. Dans *Proceedings of CLA*.

Krogel, M.-A. (2005). On propositionalization for knowledge discovery in

relational databases.

Kryszkiewicz, M. (1998). Representative association rules. Dans *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 198–209. Springer.

Kryszkiewicz, M. (2002). Concise Representations of Association Rules. In *Pattern Detection and Discovery*, volume 2447 92–109. Springer Berlin Heidelberg.

Kuperman, V., Stadthagen-Gonzalez, H. et Brysbaert, M. (2012). Age-of-acquisition ratings for 30,000 english words. *Behavior Research Methods*, *44*(4), 978–990.

Lachiche, N. (2010). *Propositionalization*, Dans C. Sammut et G. I. Webb (dir.). *Encyclopedia of Machine Learning*, (p. 812–817). Springer US : Boston, MA

Le Floc'h, A., Fisette, C., Missaoui, R., Valtchev, P. et Godin, R. (2003). Jen : un algorithme efficace de construction de générateurs pour l'identification des règles d'association. *Numéro spécial de la revue des Nouvelles Technologies de l'Information*, *1*(1), 135–146.

Lehmann, J. et Hitzler, P. (2007a). Foundations of refinement operators for description logics. Dans *International Conference on Inductive Logic Programming*, 161–174. Springer.

Lehmann, J. et Hitzler, P. (2007b). A refinement operator based learning algorithm for the alc description logic. Dans *International Conference on Inductive Logic Programming*, 147–160. Springer.

Lehmann, J. et Hitzler, P. (2010). Concept learning in description logics using refinement operators. *Machine Learning*, *78*(1-2), 203.

lod-cloud.net (2019). The linked open data cloud. https://lod-cloud.net/. Accessed : 2020-02-15.

longlivetheux (2005). Dikw pyramid : data, information, knowledge, and wisdom. [Online ; accessed 1-March-2020]. Récupéré de https://upload.wikimedia.org/wikipedia/commons/0/06/DIKW_Pyramid.svg

Luxenburger, M. (1991). Implications partielles dans un contexte. *Mathématiques et Sciences Humaines*, *113*, 35–55.

MacWhinney, B. (2000). *The CHILDES Project : Tools for Analyzing Talk*. Mahwah, NJ : Lawrence Erlbaum Associates, third edition.

Maedche, A. et Staab, S. (2000). The text-to-onto ontology learning

environment. Dans *Software Demonstration at ICCS-2000-Eight International Conference on Conceptual Structures*, volume 38, 890–930. sn.

Masolo, C., Borgo, S., Gangemini, A., Guarino, N., Oltramari, A. et Schneider, L. (2003). The wonderweb library of fundational ontologies and the dolce ontology. wonderweb deliverable d18, final report (vr. 1.0. 31-12-2003). *The WonderWeb Library of Fundational Ontologies and the DOLCE ontology. WonderWeb Deliverable D18, Final Report (vr. 1. 0. 31-12-2003).*

Mehri, R. et Valtchev, P. (2017). Mining schema knowledge from linked data on the web. Dans *International Conference on Knowledge Science, Engineering and Management*, 261–273. Springer.

Mezni, H. et Kbekbi, M. (2019). Reusing process fragments for fast service composition : a clustering-based approach. *Enterprise Information Systems*, *13*(1), 34–62.

Miralles, A., Molla, G., Huchard, M., Nebut, C., Deruelle, L. et Derras, M. (2015). Class model normalization outperforming formal concept analysis approaches with aoc-posets. Dans *CLA : Concept Lattices and their Applications*, volume 1466, 111–122.

Mistry, U. et Thakkar, A. R. (2014). Link-based classification for multi-relational database. Dans *International Conference on Recent Advances and Innovations in Engineering (ICRAIE-2014)*, 1–6. IEEE.

Moha, N., Hacene, A. M. R., Valtchev, P. et Guéhéneuc, Y.-G. (2008). Refactorings of design defects using relational concept analysis. In *International Conference on Formal Concept Analysis* 289–304.

Narasimha, V. *et al.* (2011). Liddm : A data mining system for linked data. Dans *Workshop on Linked Data on the Web. CEUR Workshop Proceedings*, volume 813, p. 108.

Nergiz, M. E., Clifton, C. et Nergiz, A. E. (2008). Multirelational k-anonymity. *IEEE Transactions on Knowledge and Data Engineering*, *21*(8), 1104–1117.

Newell, A., Shaw, J. C. et Simon, H. A. (1959). Report on a general problem solving program. Dans *IFIP congress*, volume 256, p. 64. Pittsburgh, PA.

Nica, C. (2017). *Exploring sequential data with relational concept analysis.* (Thèse de doctorat). Université de Strasbourg.

Nica, C., Braud, A., Dolques, X., Huchard, M. et Le Ber, F. (2016a). Extracting hierarchies of closed partially-ordered patterns using relational

concept analysis. Dans *International Conference on Conceptual Structures*, 17–30. Springer.

Nica, C., Braud, A., Dolques, X., Huchard, M. et Le Ber, F. (2016b). L'analyse relationnelle de concepts pour la fouille de données temporelles-application à l'étude de données hydroécologiques. 267–278.

Nica, C., Braud, A. et Le Ber, F. (2017). Hierarchies of weighted closed partially-ordered patterns for enhancing sequential data analysis. Dans *International Conference on Formal Concept Analysis*, 138–154. Springer.

Nica, Braud, D. et al. (2016). Exploring temporal data using relational concept analysis : An application to hydroecology. Dans *13th International Conference on Concept Lattices and Their Applications (CLA 2016)*, volume 1624, 299–311.

Nijssen, S. et Kok, N. (2004). Frequent graph mining and its application to molecular databases. Dans *2004 IEEE International Conference on Systems, Man and Cybernetics*, volume 5, 4571–4577. IEEE.

Nourine, L. et Raynaud, O. (1999). A fast algorithm for building lattices. *Information processing letters*, *71*(5-6), 199–204.

Ogden, C. K. (1930). Basic English : A general introduction with rules and grammar, paul treber & co. *Ltd. London*, *1940*.

Ogden, C. K. (2018). Ogden's basic english. Récupéré le 30 avril 2018 de http://ogden.basic-english.org/wordmenu.html

Ogihara, Z. P., Zaki, M., Parthasarathy, S., Ogihara, M. et Li, W. (1997). New algorithms for fast discovery of association rules. Dans *In 3rd Intl. Conf. on Knowledge Discovery and Data Mining*. Citeseer.

O'Neil, C. (2016). *Weapons of math destruction : How big data increases inequality and threatens democracy*. Broadway Books.

Pfaltz, J. L. et Taylor, C. M. (2002). Scientific knowledge discovery through iterative transformation of concept lattices. Dans *SIAM Workshop on Discrete Math. and Data Mining, Arlington, VA, USA*.

Piatetsky-Shapiro, G., Fayyad, U. et Smith, P. (1996). From data mining to knowledge discovery : An overview. *Advances in knowledge discovery and data mining*, *1*, 35.

Picard, O., Lord, M., Blondin-Massé, A., Marcotte, O., Lopes, M. et Harnad, S. (2013). Hidden structure and function in the lexicon. *arXiv preprint*

arXiv :1308.2428. Récupéré de https://arxiv.org/pdf/1308.2428

Poelmans, J., Kuznetsov, S. O., Ignatov, D. I. et Dedene, G. (2013). Formal concept analysis in knowledge processing : A survey on models and techniques. *Expert systems with applications*, *40*(16), 6601–6623.

Poulin, J., Massé, A. et Fonseca, A. (2018). Strategies for learning lexemes efficiently : A graph-based approach. Dans *COGNITIVE 2018 : The Tenth International Conference on Advanced Cognitive Technologies and Applications*, 18–23. ThinkMind.

Prediger, S. et Wille, R. (1999). The lattice of concept graphs of a relationally scaled context. Dans *International Conference on Conceptual Structures*, 401–414. Springer.

Quinlan, J. R. (1990). Learning logical definitions from relations. *Machine learning*, *5*(3), 239–266.

Rouane, M. H., Huchard, M., Napoli, A. et Valtchev, P. (2007). A proposal for combining formal concept analysis and description logics for mining relational data. Dans *International Conference on Formal Concept Analysis*, 51–65. Springer.

Rouane-Hacene, M. *et al.* (2013). Relational concept analysis : mining concept lattices from multi-relational data. *Annals of Mathematics and Artificial Intelligence*, *67*(1), 81–108. http://dx.doi.org/10.1007/s10472-012-9329-3. Récupéré de http://dx.doi.org/10.1007/s10472-012-9329-3

Rouane-Hacene, M., Fennouh, S., Nkambou, R. et Valtchev, P. (2010). Refactoring of ontologies : Improving the design of ontological models with concept analysis. Dans *2010 22nd IEEE International Conference on Tools with Artificial Intelligence*, volume 2, 167–172. IEEE.

Rouane-Hacene, M., Valtchev, P. et Nkambou, R. (2011). Supporting ontology design through large-scale fca-based ontology restructuring. Dans *International Conference on Conceptual Structures*, 257–269. Springer.

Russell, S. et Norvig, P. (2002). Artificial intelligence : a modern approach.

Sabou, M., Wroe, C., Goble, C. et Mishne, G. (2005). Learning domain ontologies for web service descriptions : an experiment in bioinformatics. Dans *Proceedings of the 14th international conference on World Wide Web*, 190–198.

Schild, K. (1991). *A correspondence theory for terminological logics : Preliminary report*. Citeseer.

Semeraro, C., Lezoche, M., Panetto, H., Dassisti, M. et Cafagna, S. (2019). Data-driven pattern-based constructs definition for the digital transformation modelling of collaborative networked manufacturing enterprises. Dans *Working Conference on Virtual Enterprises*, 507–515. Springer.

Sowa, J. F. (1983). Conceptual structures : information processing in mind and machine.

Spyropoulou, E. et De Bie, T. (2011). Interesting multi-relational patterns. Dans *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, 675–684. IEEE.

Srinivasan, A., Muggleton, S. H., Sternberg, M. J. et King, R. D. (1996). Theories for mutagenicity : A study in first-order and feature-based induction. *Artificial Intelligence*, *85*(1-2), 277–299.

Szathmary, L., Valtchev, P., Napoli, A. et Godin, R. (2008). Constructing iceberg lattices from frequent closures using generators. Dans *International Conference on Discovery Science*, 136–147. Springer.

Taddeo, M. et Floridi, L. (2005). Solving the symbol grounding problem : a critical review of fifteen years of research. *Journal of Experimental & Theoretical Artificial Intelligence*, *17*(4), 419–445.

Tarjan, R. (1972). Depth-first search and linear graph algorithms. *SIAM journal on computing*, *1*(2), 146–160.

Tuomi, I. (1999). Data is more than knowledge : Implications of the reversed knowledge hierarchy for knowledge management and organizational memory. Dans *Proceedings of the 32nd Annual Hawaii International Conference on Systems Sciences. 1999. HICSS-32. Abstracts and CD-ROM of Full Papers*, 12–pp. IEEE.

Unbehauen, J., Hellmann, S., Auer, S. et Stadler, C. (2012). Knowledge extraction from structured sources. In *Search computing* 34–52. Springer.

Valtchev, P. *et al.* (2004). Formal Concept Analysis for Knowledge Discovery and Data Mining : The New Challenges. Dans *Proc. of ICFCA 2004*, volume 2961 de *LNCS*, 352–371.

Vincent-Lamarre, P., Massé, A., Lopes, M., Lord, M., Marcotte, O. et Harnad, S. (2016). The latent structure of dictionaries. *Topics in cognitive science*, *8*(3), 625–659. Récupéré de https://onlinelibrary.wiley.com/doi/pdf/10.1111/tops.12211

Vincent-Lamarre, P., Massé, A. B., Lopes, M., Lord, M., Marcotte, O. et Harnad, S. (2017). The dictionary game. Récupéré de http://lexis.uqam.ca:8080/dictGame/index.jsp

W3C (2012). Owl 2 web ontology language document overview (second edition). https://www.w3.org/TR/owl2-overview/. Accessed : 2020-02-15.

Wajnberg, M., Valtchev, P., Lezoche, M., Masse, A. B. et Panetto, H. (2019a). Concept analysis-based association mining from linked data : A case in industrial decision making. Dans *2nd International Workshop on Data meets Applied Ontologies in Open Science and Innovation, DAO-SI 2019*, volume 2518.

Wajnberg, M., Valtchev, P., Lezoche, M., Panetto, H. et Blondin Massé, A. (2019b). Mining process factor causality links with multi-relational associations. Dans *Proceedings of the 10th International Conference on Knowledge Capture*, 263–266.

Wierzbicka, A. (1972). *Semantic Primitives*. (Frankfurt/M.)Athenäum-Verl.

Wierzbicka, A. (1996). *Semantics : Primes and universals : Primes and universals*. Oxford University Press, UK.

Wierzbicka, A. (2020). Natural semantic metalanguage. Récupéré le January 14th, 2020 de https://intranet.secure.griffith.edu.au/schools-departments/natural-semantic-metalanguage/what-is-nsm

Wille, R. (1982). *Restructuring Lattice Theory : An Approach Based on Hierarchies of Concepts*, Dans I. Rival (dir.). *Ordered Sets : Proceedings of the NATO Advanced Study Institute held at Banff, Canada, August 28 to September 12, 1981*, (p. 445–470). Springer Netherlands : Dordrecht

Wille, R. (1997). Conceptual graphs and formal concept analysis. Dans *International Conference on Conceptual Structures*, 290–303. Springer.

Wille, R. (2002). Existential concept graphs of power context families. Dans *International Conference on Conceptual Structures*, 382–395. Springer.

Yan, X. et Han, J. (2002). gSpan : Graph-based substructure pattern mining. Dans *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, 721–724. IEEE.

Ye, N. (2003). *The handbook of data mining*. CRC Press.

Zaki, M. J. (2000). Generating non-redundant association rules. Dans *Proceedings of the sixth ACM SIGKDD international conference on*

Knowledge discovery and data mining, 34–43.

Zaki, M. J. et Hsiao, C.-J. (2002). Charm : An efficient algorithm for closed itemset mining. Dans *Proceedings of the 2002 SIAM international conference on data mining*, 457–473. SIAM.

Zeleny, M. (2005). *Human systems management : Integrating knowledge, management and systems*. World Scientific.

Zeng, Q., Patel, J. M. et Page, D. (2014). Quickfoil : Scalable inductive logic programming. *Proceedings of the VLDB Endowment*, *8*(3), 197–208.

Zins, C. (2007). Conceptual approaches for defining data, information, and knowledge. *Journal of the American society for information science and technology*, *58*(4), 479–493.