UNIVERSITÉ de STRASBOURG

ÉCOLE DOCTORALE 269 : Mathématiques, Sciences de l'Information et de l'Ingénieur (MSII)

Laboratoire des sciences de l'ingénieur, de l'informatique et de l'imagerie (UMR 7357 - ICube)

THÈSE présentée par :

Vanessa Laure FOKOU

soutenue le : 17 décembre 2025

pour obtenir le grade de : **Docteur de l'Université de Strasbourg**

Discipline : Informatique

Comparaison empirique et théorique d'approches en analyse de concepts formels pour les données multi-relationnelles

THÈSE dirigée par :

Mme LE BER Florence	Directrice de Recherche, ENGEES Strasbourg
M. FERRÉ Sébastien	Professeur, Université de Rennes

ENCADRANTS :

Mme CELLIER Peggy	Maître de conférences HDR, INSA Rennes
M. DOLQUES Xavier	Maître de conférences, Université de Strasbourg

RAPPORTEURS :

Mme BERTET Karell	Professeur, La Rochelle Université
Mme MISSAOUI Rokia	Professeur, Université du Québec en Outaouais

---

AUTRES MEMBRES DU JURY :

Mme HUCHARD Marianne	Professeur, Université de Montpellier
M. STUMME Gerd	Professeur, University of Kassel

Résumé

Avec la numérisation croissante des activités humaines, une grande variété de données relationnelles devient disponible pour l'analyse. Ces données multi-relationnelles nécessitent des méthodes capables de prendre en compte simultanément les objets et les relations qui les relient, afin d'extraire des structures significatives et interprétables. Parmi les approches développées à cet effet, l'Analyse Relationnelle de Concepts (*Relational Concept Analysis* – RCA) et l'Analyse Conceptuelle de Graphes (Graph-FCA) occupent une place particulière. Toutes deux dérivent de l'Analyse Formelle de Concepts (AFC), une méthode mathématique de classification largement utilisée dans de nombreux domaines. Dans RCA, les données sont représentées sous forme de tables interconnectées, analogues au modèle entité–association, tandis que Graph-FCA modélise les données sous forme d'hypergraphes, où les objets sont les nœuds et les attributs sont portés par les hyperarêtes. Bien que partageant des fondements communs, RCA et Graph-FCA diffèrent dans leurs formalismes et leurs mécanismes d'analyse. Certaines études antérieures ont exploré leurs liens, notamment pour l'interprétation des résultats, mais sans offrir de comparaison globale et approfondie. Ce travail propose une étude comparative, à la fois théorique et empirique, des deux approches RCA et Graph-FCA, dans le but d'établir leurs similitudes et différences sur une base solide, et de fournir à l'analyste des repères pour choisir l'approche la plus adaptée selon la nature des données et les objectifs d'analyse. Nous comparons les deux approches dans leur cadre commun, à la fois du point de vue extensionnel, en analysant les groupes d'objets extraits, et du point de vue intensionnel, en examinant les descriptions associées à ces groupes. Nous étudions également leurs différences dans la modélisation des relations n-aires et le traitement des cycles, afin de déterminer dans quelle mesure ces divergences peuvent se révéler complémentaires et bénéfiques pour l'analyse. Enfin, une expérimentation sur un jeu de données réel, issu d'une ancienne pharmacopée arabe met en évidence les forces et les limites de chacune des deux approches.

Mots clés : Analyse Formelle de Concepts, Analyse Relationnelle de Concepts, Graph-FCA, Treillis, Fouille de données multi-relationnelles, Graphe de connaissances, Motif de graphe.

Abstract

With the increasing digitalization of human activities, a wide variety of relational data is becoming available for analysis. Such multi-relational data requires methods capable of simultaneously considering both the objects and the relationships connecting them, in order to extract meaningful and interpretable structures. Among the approaches developed for this purpose, Relational Concept Analysis (RCA) and Graph-FCA hold a particular place. Both derive from Formal Concept Analysis (FCA), a mathematical classification method widely used in many fields. In RCA, data is represented as interconnected tables, similar to the entity–relationship model, while Graph-FCA models data as hypergraphs, where objects are nodes and attributes are carried by hyperedges. Although they share common foundations, RCA and Graph-FCA differ in their formalisms and analysis mechanisms. Some previous studies have explored their connections, particularly regarding result interpretation, but without offering a comprehensive and in-depth comparison. This work presents a comparative study, both theoretical and empirical, of the two approaches RCA and Graph-FCA, with the aim of establishing their similarities and differences on a solid basis, and providing analysts with guidelines for choosing the most appropriate approach according to the nature of the data and the objectives of the analysis. We compare both approaches within their common framework, both from an extensional perspective, by analyzing the clusters of extracted objects, and from an intensional perspective, by examining the descriptions associated with these clusters. We also study their differences in modeling n-ary relations and handling cycles, in order to determine to what extent these divergences can be complementary and beneficial for the analysis. Finally, an experiment on a real dataset from an ancient Arabic pharmacopoeia highlights the strengths and limitations of each of the two approaches.

Keywords : Formal Concept Analysis, Relational Concept Analysis, Graph-FCA, Lattice, Multi-relational data mining, Knowledge graph, Graph pattern.

Remerciements

Il y a des saisons de la vie où l'on se sent fort et confiant, où l'on croit pouvoir avancer seul, peut-être parce qu'on parvient à visualiser le résultat, ou peut-être simplement parce qu'on le ressent, même sans pouvoir l'expliquer. Puis viennent d'autres saisons, comme celle d'une thèse, où tout semble incertain, où le chemin se brouille : parce qu'on ne sait pas vraiment ce que l'on cherche, et encore moins à quoi ressemblera le résultat. Dans ces moments, même l'imagination et la visualisation semblent ne plus être de notre côté ; la confiance ne suffit plus, l'espoir s'éloigne peu à peu, la motivation disparaît, et tout devient de plus en plus lourd... Mais, lorsqu'on est bien entouré, de personnes qui nous encouragent, qui croient en nous parfois davantage que nous-mêmes, qui nous aiment, nous soutiennent, et sont présentes avec intention, on finit par avancer, et parfois même par se surpasser.

Cette saison de ma vie, à la fois intense et riche, a été impactée et éclairée par l'aide et la présence de nombreuses personnes. Que vous ayez contribué de près ou de loin à la réalisation de ce travail, recevez l'expression de ma profonde gratitude.

Merci à mes enseignants de Master 2 de l'université de Yaoundé I, qui m'ont initiée à la recherche, et tout particulièrement Maurice Tchuenté, Norbert Tsopze et Paulin Melatagia, pour le rôle qu'ils ont joué dans mon choix de poursuivre en thèse.

Merci Florence Le Ber, Sébastien Ferré, Peggy Cellier et Xavier Dolques pour la direction et l'encadrement de ma thèse. Votre patience, votre disponibilité, votre rigueur ainsi que votre apport scientifique et méthodologique m'ont été très précieux. Merci également pour votre bienveillance et pour vos conseils ; je me suis sentie écoutée et soutenue à vos côtés. Merci Florence, pour tous les voyages effectués ensemble lors des conférences.

Merci à Karell Bertet et Rokia Missaoui d'avoir accepté de rapporter ma thèse. Merci à Marianne Huchard et Gerd Stumme d'avoir accepté d'examiner mon travail. Vos retours et questions lors de la soutenance ont été très riches et m'ont permis d'envisager mon travail sous différents angles.

Merci à Corinne Grac et Marianne Huchard d'avoir accepté d'être membres de mon CSI. Nos échanges lors des différentes réunions m'ont permis de voir d'autres aspects de mon travail et de m'auto-évaluer à chaque fois.

Merci à tous les membres de l'ANR SmartFCA, de m'avoir écouté présenter mon travail lors des différentes réunions. Vos retours et vos questions m'ont été très précieux.

Mes remerciements vont également aux membres de l'équipe SDC du laboratoire ICube, pour tous les moments partagés. Réunions et barbecues d'équipe, repas au RU, échanges dans les couloirs ou dans les bureaux : rien n'aurait été pareil sans vous.

Je remercie ma famille pour son amour inconditionnel, son soutien sans faille, ses prières et ses messages d'encouragement. Merci Gisèle d'être venue me chercher à l'aéroport lors de mon arrivée — je me serais sûrement perdue sinon. Merci pour tout ce que nous partageons au quotidien. Merci Loth et Yolande pour tous les moments chaleureux passés chez

vous ; c'est toujours un plaisir de venir à la maison.

Un grand merci à Farida, Raïssa, Kisito, Stéphane, Durande, Audrey, Gwladys, Charnelle, Roblex, Quentin, Roméo, Arsène, Chris, Cabrel, Marie-Ange. Qu'il s'agisse de nos longues discussions au téléphone, des moments partagés ensemble, ou de vos messages d'encouragement et de motivation, vous avez contribué à garder mon moral au plus haut. Merci à Arsène, Cabrel, Chris et Marie-Ange pour toutes les soirées passées ensemble à jouer, rire, manger, danser et boire. Je suis ravie de vous avoir rencontrés. Merci Gwladys de m'avoir accueillie durant toutes mes missions à Rennes, car une semaine avec toi vaut bien mieux qu'un séjour dans un hôtel chic. Merci Audrey pour ces soirées passées à nous tresser les cheveux, regarder des films et rire aux éclats. Merci Raïssa de m'avoir aidée à préparer ma première présentation de conférence en anglais, pour tes corrections de prononciation et pour ta grande patience. Merci Roméo pour ta présence, ton amour et ton soutien infaillible. Merci de croire en moi et de toujours m'encourager à me réaliser pleinement.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

Avec la numérisation croissante des activités humaines, un volume considérable de données devient disponible pour l'analyse dans de nombreux domaines comme par exemple l'hydro-écologie [Dolques et al., 2021 ; Nica et al., 2016b] ou la médecine [Rouane-Hacene et al., 2009]. Ces données issues de la transformation numérique se caractérisent par leur volumétrie, leur richesse sémantique, leur complexité structurelle et leur nature multi-relationnelle, car elles décrivent une grande variété d'entités et les interactions entre elles. Les données multi-relationnelles, qu'elles soient spatiales, temporelles, ou décrivant des liens entre individus, peuvent être efficacement représentées sous forme de graphes. Par exemple, dans le domaine environnemental, les graphes sont utilisés pour modéliser les interactions sociales entre les animaux. Avec l'essor du Web sémantique [Hitzler et al., 2009], ces représentations se généralisent à travers les graphes de connaissances, tels que des graphes RDF [Hitzler et al., 2009] ou des graphes conceptuels [Chein et Mugnier, 2008 ; Sowa, 1984], qui permettent de représenter des informations complexes comme des ensembles d'entités reliées par des relations binaires, voir n-aires. La représentation et l'analyse de ces structures relationnelles sont ainsi devenues essentielles pour la compréhension de phénomènes complexes impliquant de multiples entités.

Les méthodes traditionnelles d'analyse de données, telles que les statistiques descriptives, s'avèrent insuffisantes pour mettre en évidence les connaissances cachées et comprendre les phénomènes sous-jacents à des données complexes. Des approches d'apprentissage automatique [Scarselli et al., 2008] ont été proposées et produisent de bons résultats, notamment pour des tâches de classification. Toutefois, ces méthodes nécessitent généralement de grands volumes de données et génèrent des modèles dont l'interprétation par des experts métier peut être difficile. Or, dans certains domaines tels que les sciences humaines et sociales, il est essentiel de pouvoir fournir des résultats compréhensibles et interprétables par les spécialistes du domaine. Les approches symboliques [Agrawal et al., 1993], telles que l'Analyse Formelle de Concepts (AFC) [Ganter et Wille, 1999], s'avèrent particulièrement adaptées à ces situations, car elles permettent d'extraire des structures interprétables et exploitables, même à partir de volumes de données relativement modestes. En effet, l'AFC est centrée sur l'utilisateur et offre un support pour visualiser et interagir avec les données et les motifs découverts, facilitant ainsi l'analyse.

L'Analyse Formelle de Concepts (AFC) [Ganter et Wille, 1999] est une méthode mathématique qui s'appuie sur la théorie des treillis [Barbut et Monjardet, 1970 ; Birkhoff, 1940] pour la découverte de connaissances, la classification et l'analyse de données en gé-

néral. L'AFC constitue une méthode de découverte de connaissance à part entière et est largement utilisée pour l'analyse de données dans de nombreux domaines [Ferré et al., 2020 ; Poelmans et al., 2013]. Initialement conçue pour les données tabulaires, l'application directe de l'AFC à des données plus complexes comme des données relationnelles présente des limites, car elle nécessite des modélisations supplémentaires sur les données, ce qui peut entraîner une perte d'informations structurelles et compliquer l'interprétation des motifs extraits. Ces contraintes ont motivé le développement d'extensions de l'AFC permettant d'utiliser des descriptions d'objets plus complexes que de simples ensembles d'attributs binaires [Ferré et Ridoux, 2000 ; Ganter et Kuznetsov, 2001 ; Kaytoue et al., 2015]. Plus particulièrement, des méthodes capables d'explorer directement des données multi-relationnelles [Džeroski, 2010] ont été proposées, afin de capturer les structures relationnelles pertinentes sans perdre d'informations. En conséquence, l'AFC a été étendue aux données multi-dimentionnelles [Lehmann et Wille, 1995 ; Voutsadakis, 2002], ainsi qu'aux données multi-relationnelles [Bazin et al., 2024 ; Ferré, 2015 ; Kötters, 2013 ; Rouane-Hacene et al., 2013 ; Wille, 1997].

Comme on peut le constater, plusieurs extensions de l'AFC ont été proposéés, y compris des développements récents [Bazin et al., 2024 ; Ferré, 2023] visant à prendre en compte les données complexes et multi-relationnelles. De même, l'inventaire en ligne des logiciels de l'AFC [1] montre que de nombreux outils implémentent l'AFC et ses extensions, par exemple Conexp-Clj [Hanika et Hirth, 2019], RCAExplore [Dolques et al., 2019], FCA4J [Gutierrez et al., 2022], GFCA [Ferré, 2019] ou encore GALACTIC [Demko et al., 2022]. Cependant, aucun de ces outils ne constitue une plateforme consensuelle et centralisée offrant une bibliothèque d'algorithmes interopérables pour l'analyse de données réelles dans des domaines complexes comme la biologie, la chimie ou la médecine. C'est dans ce contexte qu'intervient le projet **ANR SmartFCA**, qui regroupe cinq équipes françaises travaillant dans le domaine de l'AFC. L'objectif de ce projet est de contribuer au développement de l'AFC et de ses extensions, tout en fournissant une plateforme opérationnelle et générique rassemblant les variantes de l'AFC pour l'analyse des données provenant du monde réel.

Cette thèse s'inscrit dans la partie du projet SmartFCA relative à l'étude des extensions relationnelles de l'AFC. Elle se concentre plus particulièrement sur une analyse comparative de l'Analyse Relationnelle de Concepts [Rouane-Hacene et al., 2013] (*Relational Concept Analysis* (RCA), en anglais) et de l'analyse conceptuelle de graphes (Graph-FCA/GCA) [Ferré, 2015]. Dans RCA, les données relationnelles sont représentées par un ensemble de tables (contextes) objets-attributs et objets-objets. En revanche, GCA modélise les données sous la forme d'un hypergraphe, où les objets correspondent aux nœuds et les attributs sont portés par des hyper-arêtes. Certains liens entre RCA et les graphes ont été étudiés pour les données temporelles [Nica et al., 2016a ; Nica et al., 2020]. De même, quelques études se sont penchées sur le rapprochement et la comparaison entre RCA et GCA [Ferré et Cellier, 2018 ; Keip et al., 2020], mais elles se sont limitées à des aspects spécifiques, tels que l'interprétation des résultats, sans proposer une comparaison appro-

1. https://upriss.github.io/fca/fcasoftware.html

fondie des deux approches. Il est donc essentiel d'établir une base solide de comparaison entre les deux approches, afin de mieux caractériser, tant sur le plan théorique que pratique, leurs similitudes et leurs différences. Un tel cadre comparatif vise à fournir à l'analyste des repères pour choisir l'approche la plus adaptée, en fonction de la nature des données à traiter et des résultats attendus.

Dans cette thèse, nous présentons nos contributions portant sur une comparaison empirique et théorique des deux approches RCA et GCA. Bien que leurs objectifs et leurs résultats semblent similaires, ces approches diffèrent sur plusieurs aspects, notamment la définition des concepts. Nous nous sommes intéressés à deux axes de comparaisons. Tout d'abord, nous avons comparé leurs similitudes, à la fois du point de vue extensionnel, c'est-à-dire en comparant les groupes d'individus qui sont extraits par les deux approches, et du point de vue intensionnel, en examinant les descriptions associées à ces groupes. Ensuite, nous avons comparé les deux méthodes du point de vue de leurs différences. Ainsi, nos contributions se repartissent comme suit.

1. **Comparaison extensionnelle de RCA et GCA**. Nous commençons par comparer RCA et GCA du point de vue des extensions de concepts. À travers plusieurs exemples, nous montrons que l'ensemble des extensions des concepts de RCA est inclus dans celui de GCA. Autrement dit, pour tout concept RCA défini sur un jeu de données, il existe un concept GCA possédant la même extension. Ce constat empirique est appuyé par des démonstrations théoriques qui établissent formellement cette inclusion.
2. **Comparaison intensionnelle de RCA et GCA**. Nous poursuivons par la comparaison des intensions des concepts et démontrons que l'ensemble des intensions des concepts RCA est inclus dans celui des intensions des concepts GCA. Ces démonstrations reposent sur une étape préalable de modélisation des intensions dans une représentation commune (sous forme de graphes) car les intensions de concepts des deux approches ne sont pas directement comparables. Combinée au résultat de la comparaison extensionnelle, cette comparaison intensionnelle permet de conclure que sur un même jeu de données, l'ensemble des concepts RCA est inclus dans celui des concepts GCA, ce qui illustre que GCA est plus expressif que RCA.
3. **Comparaison de RCA et GCA à travers leurs différences**. Nous avons conduit une étude comparative centrée sur leurs différences, afin d'examiner dans quelle mesure ces divergences peuvent être complémentaires et bénéfiques pour l'analyse. Nous avons notamment examiné les distinctions dans la modélisation des relations n-aires et le traitement des cycles par chacune des approches. Une mise en œuvre sur un jeu de données réel, issu d'une ancienne pharmacopée arabe [Kahl, 2009], a permis de mettre en évidence les forces et les limites pratiques de RCA et GCA.

Ce manuscrit est organisé de la manière suivante.

— La première partie présente le cadre général de ce travail. Le chapitre 2 introduit

l'Analyse Formelle de Concepts (AFC), ainsi qu'un aperçu de ses principales extensions destinées aux données multi-relationnelles. Le chapitre 3 présente l'Analyse Relationnelle de Concepts (RCA). Le chapitre 4 est consacré à l'analyse conceptuelle des graphes (Graph-FCA/GCA). La première partie s'achève par le chapitre 5, qui résume l'état de l'art des travaux abordant, de manière directe ou indirecte, les liens entre RCA et GCA.

— La deuxième partie, consacrée à nos contributions, débute par la présentation de notre approche méthodologique (chapitre 6). Le chapitre 7 traite de la comparaison extensionnelle de RCA et GCA. Le chapitre 8 est consacré à la comparaison intensionnelle de ces deux approches. Le chapitre 9 aborde la comparaison de RCA et GCA à travers leurs différences.

— Enfin, le chapitre 10 conclut et propose plusieurs perspectives d'approfondissement pour ce travail.

Première partie

Cadre général

Chapitre 2

Analyse Formelle de Concepts

Sommaire

2.1 Introduction

L'Analyse Formelle de Concepts (AFC) (en anglais *Formal Concept Analysis*) est un formalisme introduit par Wille en 1982 [Wille, 1982], puis dans les travaux de Ganter et Wille [Ganter et Wille, 1999], permettant la classification conceptuelle d'un ensemble d'*objets* (appelés parfois *individus*) décrits par des *attributs*. L'AFC est basée sur la théorie des treillis [Birkhoff, 1940] et des treillis de Galois [Barbut et Monjardet, 1970].

L'AFC a largement été appliquée dans divers domaines comme outil d'analyse et d'exploration des données. En médecine, l'AFC a été utilisée pour identifier les combinaisons de réactions médicamenteuses et les interactions médicamenteuses [Rouane-Hacene et al., 2009 ; Villerd et al., 2010]. Elle a également été utilisée pour identifier les dépendances entre les données démographiques et le degré d'activité physique [Belohlavek et al., 2011 ; Sklenar et al., 2005]. On retrouve également plusieurs applications de l'AFC dans l'exploration du web (*web mining*) [Bruno et al., 2005 ; Ebner et al., 2010 ; Eklund et al., 2004], en bio-informatique [Alam et al., 2012 ; Kaytoue et al., 2009 ; Keller et al., 2012], en linguistique [Falk et al., 2010 ; Priss et Old, 2004], en chimie [Lounkine et al., 2008 ; Stumpfe et al., 2010] et plus encore. La revue de la littérature [Poelmans et al., 2013] recense un grand nombre des applications de l'AFC dans les domaines sus-évoqués et dans d'autres domaines.

Dans ce chapitre, nous présentons les définitions et concepts fondamentaux de l'AFC, ainsi qu'un aperçu de ses principales extensions. La section 2.2 introduit les notions mathématiques relatives à la théorie des treillis, suivie, en section 2.3, par la présentation des notions de base de l'AFC. La section 2.4 décrit deux algorithmes classiques de l'AFC, tandis que la section 2.5 présente une synthèse des extensions de l'AFC appliquées aux données complexes et multi-relationnelles.

2.2 Quelques éléments de la théorie des treillis

Dans cette section, nous présentons certaines notions fondamentales de la théorie des treillis en s'inspirant de [Barbut et Monjardet, 1970].

2.2.1 Ensemble partiellement ordonné

Soit E un ensemble, on appelle relation binaire R sur E, toute relation de $E \times E$. Il s'agit d'un ensemble de couples $(x, y) \in E \times E$. Pour $x, y \in E$, on écrit en notation infixe, xRy pour indiquer que $(x, y) \in R$, et $x \not R\, y$ sinon.

Définition 2.1 (Ordre ou ordre partiel). Soient E un ensemble et R une relation bi-

naire sur E. Un ordre partiel sur E donné par la relation R vérifie les propriétés suivantes, $\forall x, y, z \in E$:

- réflexivité : xRx,
- antisymétrie : si xRy et yRx alors $x = y$,
- transitivité : si xRy et yRz alors xRz.

Exemple : L'ordre d'inclusion des parties d'un ensemble est un ordre partiel.

Définition 2.2 (Ensemble partiellement ordonné). Un ensemble partiellement ordonné (*partially ordored set (poset)*) est un couple (E, R) où E est un ensemble et R est une relation d'ordre sur E.

Un ordre R est qualifié de *total* et non de *partiel* si deux éléments x et y de E sont toujours comparables i.e $\forall(x, y)$ de E où $x \neq y$ on a xRy ou yRx.

Exemple : L'ordre des points de la gauche à la droite sur une droite orientée est un ordre total.

On utilise souvent le symbole $\leq$ pour les relations d'ordre et $x \leq y$ se lit "x *est inférieur ou égal à* y". À toute relation d'ordre $\leq$, on associe sa relation d'ordre strict notée $<$, définie par $x < y$ si $x \leq y$ et $x \neq y$. Elle se déduit de la relation $\leq$ en remplaçant la propriété de réflexivité, par celle d'irréflexivité, c'est-à-dire $x \not R\, x\ \forall x \in E$.

2.2.2 Diagramme de Hasse d'une relation d'ordre

Définition 2.3 (Majorant, minorant). Soient $(E, \leq)$ un ensemble ordonné et $x, y \in E$, on dit que y est un majorant de x si et seulement si $x \leq y$. Inversement, on dira que y est un minorant de x si et seulement $y \leq x$.

Définition 2.4 (Successeur, prédécesseur). Soient $(E, \leq)$ un ensemble ordonné et $x, y \in E$. On dit que x succède à y (x couvre y) et on note $x \succ y$, si et seulement si x est un majorant de y, et tel qu'il n'y ait aucun élément intermédiaire entre x et y, i.e il n'existe aucun élément $z \in E$ tel que $y < z < x$. De façon duale, y précède x ($y \prec x$) et on dit que y est couvert par x.

Tout ensemble ordonné $(E, \leq)$ peut être représenté graphiquement par un diagramme appelé "diagramme de Hasse" basé sur la relation de *couverture*. Dans ce diagramme, les nœuds représentent les éléments de E, et les arcs indiquent la relation de couverture entre ces éléments. À partir d'un tel diagramme, on peut lire la relation d'ordre comme suit : $x < y$ si et seulement si x se trouve en dessous de y en suivant un chemin dans le diagramme. Ce diagramme s'élabore de la manière suivante :

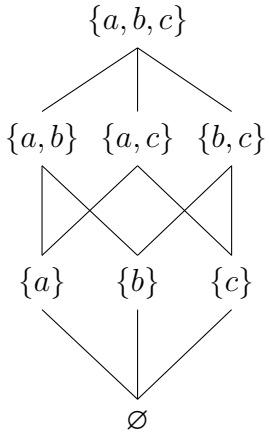

Figure 2.1 – Diagramme de Hasse de $(\mathcal{P}(X), \subseteq)$ avec $X = \{a, b, c\}$.

- Tout élément de E est représenté par un point dans le plan.
- Si $x, y \in E$ et $x \prec y$ alors le point correspondant à y doit être au-dessus de celui correspondant à x et les deux points sont reliés par un segment.
- Par contrainte de lisibilité, les arcs réflexifs et les arcs de transitivité ne sont pas représentés.

Exemple : Soit l'ensemble $X = \{a, b, c\}$, et $(\mathcal{P}(X), \subseteq)$ l'ensemble des parties de X ordonné par inclusion, le diagramme de Hasse de $(\mathcal{P}(X), \subseteq)$ est donné à la figure 2.1. On remarque par exemple que $\{a\} \prec \{a, b\}$, i.e. $\{a, b\}$ couvre $\{a\}$.

Définition 2.5 (Supremum, infimum). Soient $(E, \leq)$ un ensemble ordonné et $A \subset E$ une sous-partie de E. L'ensemble des majorants de A est l'ensemble $S = \{y \in E \mid \forall x \in A, y \geq x\}$ et l'ensemble des minorants de A est l'ensemble $I = \{y \in E \mid \forall x \in A, y \leq x\}$. S'il existe un plus petit élément dans l'ensemble S des majorants de A, il est appelé $supremum$ ou borne supérieure de A et on le note $sup(A)$ ou $\bigvee A$. Dualement, le plus grand élément de l'ensemble I des minorants de A est appelé $infimum$ ou borne inférieure de A et on le note $inf(A)$ ou $\bigwedge A$.

2.2.3 Treillis

Il existe dans la littérature deux définitions pour un treillis : une définition *algébrique* [Birkhoff, 1940] et une définition *relationnelle* [Barbut et Monjardet, 1970]. La définition relationnelle (encore dite *combinatoire*) fait intervenir les propriétés de la relation d'ordre définie sur l'ensemble tandis que la définition algébrique fait intervenir les propriétés d'opérations définies sur l'ensemble.

Définition 2.6 (Treillis algébrique). Un treillis est un ensemble E muni de deux opérations notées $\vee$ et $\wedge$ et vérifiant les axiomes suivants, $\forall x, y, z \in E$:

- *idempotence :* $x \vee x = x \; et \; x \wedge x = x$
- *commutativité :* $x \vee y = y \vee x \; et \; x \wedge y = y \wedge x$
- *associativité :* $(x \vee y) \vee z = x \vee (y \vee z) \; et \; (x \wedge y) \wedge z = x \wedge (y \wedge z)$
- *absorption :* $x \wedge (x \vee y) = x \; et \; x \vee (x \wedge y) = x$

Définition 2.7 (Treillis relationnel). Un treillis est un ensemble ordonné E tel que chaque couple (x, y) d'éléments possède un supremum $(x \vee y)$ et un infimum $(x \wedge y)$. Il est dit *sup-demi-treillis* dans le cas où seulement le *supremum* existe. Inversement, il est dit *inf-demi-treillis* si seulement l'*infimum* existe.

Donc un treillis est à la fois un *sup-demi-treillis* et un *inf-demi-treillis*. Un treillis est dit *complet* si le supremum $\bigvee X$ et l'infimum $\bigwedge X$ existent pour toute partie X de E. Un treillis complet possède un plus grand élément $\bigvee E$ noté top $(\top)$ et un plus petit élément $\bigwedge E$ noté $bottom$ $(\bot)$.

Exemple : L'ensemble des parties de E ordonnées par inclusion $(\mathcal{P}(E), \subseteq)$ présenté à la figure 2.1 par un diagramme de Hasse est un treillis.

2.2.4 Opérateurs de fermeture et connexion de Galois

Définition 2.8 (Opérateur de fermeture). Soit $(E, \leq)$ un ensemble ordonné. On appelle opérateur de fermeture sur l'ensemble E, toute application $h : E \to E$, qui vérifie ces trois propriétés $\forall x, y \in E$:

- $h(x) \geq x$ *: h est extensive*
- $x \geq y \Rightarrow h(x) \geq h(y)$ *: h est monotone croissante*
- $h(h(x)) = h(x)$ *: h est idempotente*

On parle aussi de fermeture de $Moore$ qu'on note souvent $h(x) = \bar{x}$. Étant donné un opérateur de fermeture h sur un ensemble ordonné $(E, \leq)$, un élément $x \in E$ est dit *fermé* si $h(x) = x$.

Définition 2.9 (Connexion de Galois). Soient $\alpha : O \to A$ et $\beta : A \to O$ deux applications entre deux ensembles ordonnés $(O, \leq_O)$ et $(A, \leq_A)$. Les applications α et β forment une connexion de Galois entre $(O, \leq_O)$ et $(A, \leq_A)$ si elles vérifient les conditions suivantes pour tous $o, o_1, o_2 \in O$ et $a, a_1, a_2 \in A$:

- $o_1 \leq_O o_2 \Rightarrow \alpha(o_2) \leq_A \alpha(o_1)$: α est décroissante

- $a_1 \leq_A a_2 \Rightarrow \beta(a_2) \leq_O \beta(a_1)$: β est décroissante
- $o \leq_O \beta(\alpha(o))$ et $a \leq_A \alpha(\beta(a))$: $\beta \circ \alpha$ et $\alpha \circ \beta$ sont extensives

Les propriétés données dans la définition précédente sont équivalente à la formule :

$$o \leq_G \beta(a) \Longleftrightarrow a \leq_A \alpha(o) \tag{2.1}$$

En d'autres termes, dire que (α, β) forme une *connexion de Galois* revient à dire que les applications $h = \beta \circ \alpha$ et $h' = \alpha \circ \beta$ obtenues par composées des applications α et β sont des *opérateurs de fermetures*.

Définition 2.10 (Fermetures de Galois). Soit (α, β) une connexion de Galois entre deux ensembles ordonnés $(O, \leq_O)$ et $(A, \leq_A)$. On appelle fermetures de Galois, les fermetures $h = \beta \circ \alpha$ dans O et $h' = \alpha \circ \beta$ dans A associées à une connexion de Galois (α, β).

2.3 Notions fondamentales de l'AFC

L'AFC vise à découvrir des descriptions conceptuelles (*concepts formels*) à partir d'un ensemble d'objets décrits par des attributs unaires (*contexte formel*). Cette section rappelle les principales notions de contexte formel, concept formel et *treillis de concepts*.

2.3.1 Contextes et concepts formels

Définition 2.11 (Contexte formel). Un contexte formel est un triplet (O, A, I), où O est un ensemble non vide d'objets, A un ensemble non vide d'attributs et I une relation binaire entre O et A ($I \subseteq O \times A$). Si nous avons $(o, a) \in I$, avec $o \in O$ et $a \in A$, on dit que l'objet o possède l'attribut a.

Un contexte peut être représenté graphiquement par un tableau de dimension $|O| \times |A|$, où chaque ligne représente un objet formel et chaque colonne représente un attribut formel. L'intersection de la ligne o avec la colonne a contient une croix si et seulement si $(o, a) \in I$.

Exemple : Le tableau 2.1 illustre un contexte formel qui décrit des personnes (objets) par leurs caractéristiques (attributs). Dans ce contexte, $O = \{Alice, Bob, Charlie, Julie\}$ et $A = \{$*male, female, countryside, single, married*$\}$ et la relation d'incidence I est identifiée par les croix.

Définissons α et β, deux opérateurs image sur l'ensemble des parties de O et l'ensemble des parties de A.

TABLEAU 2.1 – Contexte formel $\mathcal{K}_{Person}$ des personnes et leurs caractéristiques.

$\mathcal{K}_{Person}$	male	female	city	countryside	single	married
Alice		×	×		×	
Bob	×			×		×
Charlie	×		×		×	
Julie		×		×		×

Pour chaque ensemble d'objets $G \in \mathcal{P}(O)$ [1], les attributs partagés par ces objets peuvent être obtenus à l'aide de l'application $\alpha : \mathcal{P}(O) \to \mathcal{P}(A)$ définie par :

$$\alpha(G) = \{a \in A \mid \forall o \in G, (o, a) \in I\} \tag{2.2}$$

$\alpha(G)$ est l'ensemble des attributs communs d'un ensemble d'objets G.
De même l'application $\beta : \mathcal{P}(A) \to \mathcal{P}(O)$ qui associe à un ensemble d'attributs $M \subseteq A$ tous les objets partagés par ces attributs est définie par :

$$\beta(M) = \{o \in O \mid \forall a \in M, (o, a) \in I\} \tag{2.3}$$

La paire de fonctions (α, β), résumant les liaisons entre les objets et les attributs dans le contexte formel, définit une connexion de Galois entre les deux ensembles ordonnés $(\mathcal{P}(O), \subseteq)$ et $(\mathcal{P}(A), \subseteq)$ (voir la définition 2.9 de la connexion de Galois).

Exemple : Dans le contexte $\mathcal{K}_{Person}$ du tableau 2.1, on trouve les images suivantes :

- $\alpha(\{Alice\}) = \{$*female, city, single*$\}$
- $\alpha(\{Charlie\}) = \{$*male, city, single*$\}$
- $\alpha(\{Alice, Charlie\}) = \alpha(\{Alice\}) \cap \alpha(\{Charlie\}) = \{$*city, single*$\}$
- $\beta(\{$*female*$\}) = \{Alice, Julie\}$

Définition 2.12 (Concept formel). Un concept formel est un couple (A, B) avec $A \in \mathcal{P}(O)$, $B \in \mathcal{P}(A)$ tels que $\alpha(A) = B$ et $\beta(B) = A$; A et B représentant respectivement l'extension (en anglais *extent*) et l'intension (en anglais *intent*) du concept (A, B).

En d'autres termes, un concept formel est un couple (A, B) où A et B sont des ensembles fermés (au sens des fermetures de Galois, définition 2.10) et $\alpha(A) = B$ (ou de façon équivalente $\beta(B) = A$).

Exemple : Dans le contexte $\mathcal{K}_{Person}$ (tableau 2.1), $(\{Alice, Charlie\},\ \{$*city, single*$\})$ forme un concept formel, alors que $(\{Alice\},\ \{$*female*$\})$ n'en est pas un. En effet, on a :

1. $\mathcal{P}(O)$ représente l'ensemble des parties d'un ensemble O.

- (1) $\alpha(\{Alice, Charlie\})$ = {*city, single*}, (2) $\{Alice, Charlie\}$ est un ensemble fermé car, $\beta(\alpha(\{Alice, Charlie\}))$ = $\beta(\{$*city, single*$\})$ = $\{Alice, Charlie\}$, (3) {*city, single*} est fermé car, $\alpha(\beta(\{$*city, single*$\}))$ = $\alpha(\{Alice, Charlie\})$ = {*city, single*}
- $\{Alice\}$ et {*female*} sont des ensembles fermés mais, $\alpha(\{Alice\}) \neq \{$*female*$\}$

Intuitivement, un concept formel peut être visualisé comme un rectangle de croix maximal dans une permutation des lignes et des colonnes de la table. Par exemple, dans le contexte $\mathcal{K}_{Person}$, le concept $(\{Alice, Julie\}, \{$*female*$\})$ correspond au rectangle vertical de croix dans la colonne *female*.

2.3.2 Treillis de concepts

Étant donné un contexte formel, on obtient un ensemble de concepts sur lequel est défini un ordre. La relation de sous-concept/super-concept est une relation d'ordre sur l'ensemble des concepts définie pour deux concepts $C_1 = (A_1, B_1)$ et $C_2 = (A_2, B_2)$ par :

$$(A_1, B_1) \leq (A_2, B2) \Longleftrightarrow A_1 \subseteq A_2 \ (\Longleftrightarrow B_2 \subseteq B_1) \tag{2.4}$$

On dit que C_1 est un sous-concept de C_2, et C_2 un sur-concept de C_1. La relation "$\leq$" s'appuie sur deux inclusions duales, entre ensembles d'objets et entre ensembles d'attributs et peut ainsi être interprétée comme une relation de *généralisation/spécialisation* entre les concepts formels. Ainsi, on dit que C_1 est un sous-concept de C_2 et C_2 un sur-concept de C_1, c'est-à-dire que le concept C_2 est plus général que le concept C_1 (inversement, C_1 est plus spécifique que C_2).

L'ensemble ordonné de tous les concepts muni de la relation d'ordre entre ces concepts forme un treillis complet appelé le *treillis de concepts*.

Définition 2.13 (Treillis de concepts). L'ensemble $\mathcal{C}$ des concepts extraits d'un contexte formel muni de la relation de généralisation/spécialisation $\leq$ (inclusion des extensions ou inclusion inverse des intensions) forme un treillis de concepts $(\mathcal{C}, \leq)$. La relation $\leq$ est aussi dite relation de subsomption.

La figure 2.2 présente le treillis de concepts correspondant au contexte $\mathcal{K}_{Person}$ (tableau 2.1). Ce treillis a été construit avec l'outil RCAExplore [Dolques et al., 2019]. Dans cette représentation, chaque concept est représenté par une boîte de trois compartiments qui, de haut en bas représentent l'identifiant du concept, son intension et son extension. Cette convention est utilisée tout au long de ce manuscrit. Par exemple, le concept $Person\_8$ a pour extension $\{Alice, Charlie\}$ et pour intension {*single, city*}. On peut associer une description textuelle à ces concepts. $Person\_8$ représente alors les personnes qui sont célibataires (*single*) et qui vivent en ville (*city*). Dans ce treillis, pour illustrer la relation de subsumption, nous avons par exemple $Person\_4 \leq Person\_8$.

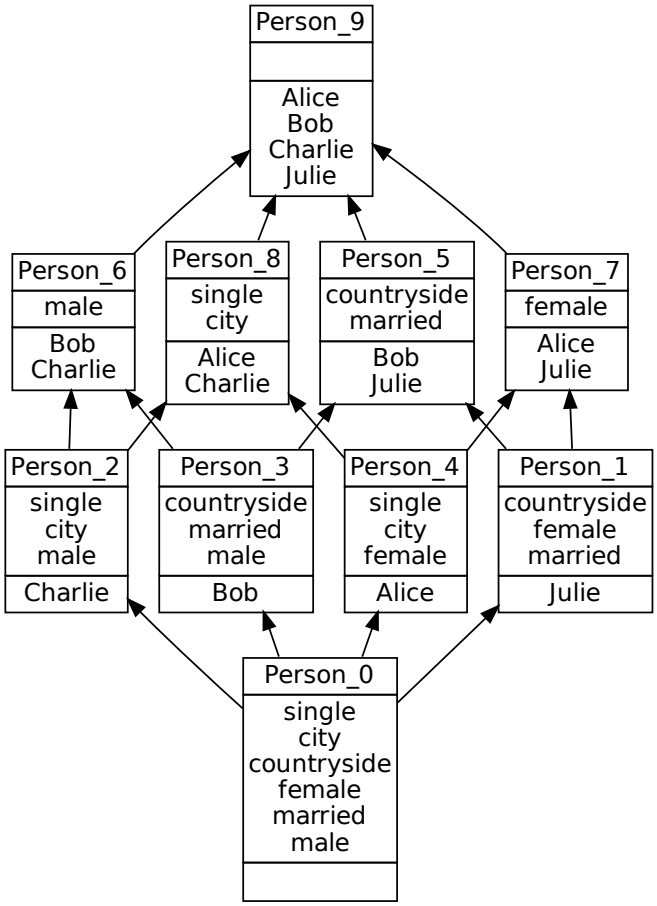

FIGURE 2.2 – Treillis de concepts du contexte $\mathcal{K}_{Person}$ (tableau 2.1).

Les concepts top ($\top$) $Person\_9$ et bottom ($\bot$) $Person\_0$ sont respectivement le supremum et l'infimum du treillis (voir définition 2.5). Ils expriment respectivement les attributs communs à tous les objets et les objets qui possèdent tous les attributs. En pratique, l'intension du $\top$ concept est dans la majorité des cas vide, car un attribut partagé par tous les objets est non discriminant de l'ensemble d'attributs. De même, l'extension du $\bot$ concept est dans la majorité des cas vide, car un objet qui possède tous les attributs est non discriminant de l'ensemble d'objets. De tels objets et attributs ne permettent pas de créer de séparations utiles dans le treillis de concepts, c'est-à-dire que les supprimer ne change rien à la structure du treillis.

Dans le treillis de la figure 2.2, on remarque que certains objets (personnes) ainsi que certains attributs (caractéristiques) apparaissent plusieurs fois. Pour illustration, $Alice$ apparaît dans les concepts $Person\_4$, $Person\_7$, $Person\_8$ et $Person\_9$. Ce genre d'affichage est un peu redondant, car si $Alice$ apparaît dans le concept $Person\_4$, et que $Person\_4$ est un sous-concept de $Person\_7$, $Person\_8$ et $Person\_9$, alors $Alice$ apparaît dans $Person\_7$, $Person\_8$ et $Person\_9$.

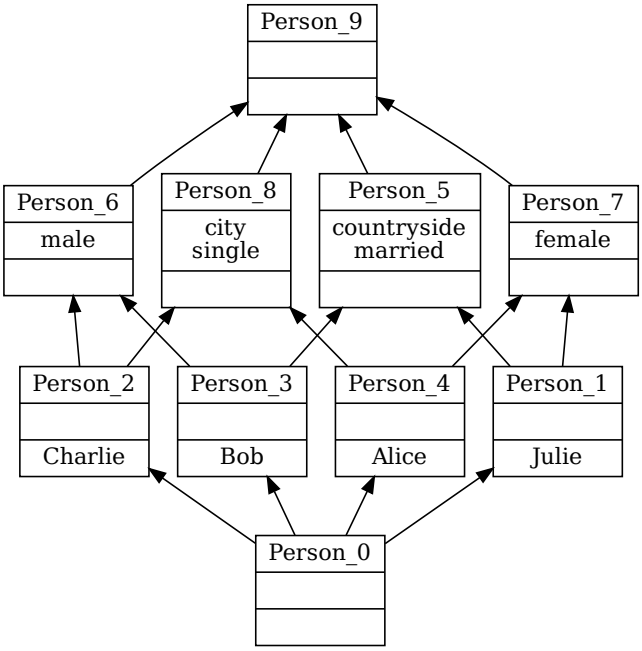

Figure 2.3 – Treillis réduit ($\mathcal{L}^0_{Person}$) correspondant au treillis de la figure 2.2.

L'affichage complet peut rendre le treillis moins lisible, surtout pour des treillis de grande taille. Une représentation permettant d'éviter ce problème consiste à supprimer les redondances d'objets et d'attributs dans les extensions et les intensions de concepts respectivement. En effet, dans un concept $C = (A, B)$, A est présent dans tous les sur-concepts de C et symétriquement, B est dans tous les sous-concepts de C. Ainsi, la suppression des redondances consiste à retirer de l'extension d'un concept (resp. de son intension) tous les objets (resp. attributs) qui apparaissent dans les extensions ses sous-concepts (resp. sur-concepts). La structure obtenue est appelée *treillis d'héritage* ou *treillis réduit (simplifié)*.

Dans cette représentation réduite, les extensions sont obtenues par héritage ascendant des objets et les intensions par héritage descendant des attributs. En d'autres termes, dès qu'un objet apparaît dans un concept, il est hérité par ses sur-concepts et dès qu'un attribut apparaît dans un concept il est hérité par ses sous-concepts. À titre d'exemple, la figure 2.3 montre le treillis correspondant à la représentation réduite du treillis de la figure 2.2. Notons que les treillis réduits peuvent toutefois devenir moins pratiques quand le nombre de concepts est élevé, car il faut alors naviguer dans le treillis pour connaître tous les éléments situés dans l'extension et l'intension d'un concept donné. Dans le reste de ce manuscrit, sauf mention contraire, la représentation réduite du treillis sera systématiquement utilisée.

2.3.3 Analyse des treillis de concepts

En pratique, la construction de treillis sur des jeux de données volumineux peut engendrer un nombre très élevé de concepts, ce qui complique l'extraction des informations pertinentes.

Pour améliorer la lisibilité et réduire la complexité du treillis complet, deux structures dérivées sont couramment utilisées : le *sup-demi-treillis* ou treillis *Iceberg* [Stumme et al., 2001, 2002], et les sous-hiérarchies de Galois [Dicky et al., 1994 ; Godin et Mili, 1993], encore appelées *AOC-poset* (*Attribute-Object-Concept poset*) [Osswald et Petersen, 2003 ; Petersen, 2001].

Un treillis Iceberg d'un treillis est obtenu en filtrant les concepts pour ne garder que ceux qui vérifient la condition de fréquence par rapport à un support [2] seuil donné. Ainsi, le treillis iceberg inclut les concepts les plus généraux et exclut ceux qui sont plus spécifiques.

Les sous-hiérarchies de Galois ne conservent dans un treillis que les concepts introduisant un objet (concepts objets) ou un attribut (concepts attributs). En d'autres termes, un concept $C = (A, B)$ est conservé si et seulement si $\exists o \in A$ tel qu'aucun des sous-concepts de C ne contient o dans son extension ou $\exists a \in B$ tel qu'aucun des sur-concepts de C ne contient a dans son intension. L'inconvénient est que la structure n'est plus nécessairement un treillis et que certaines abstractions potentiellement utiles sont éliminées. Par contre, le nombre de concepts dans une sous-hiérarchie peut-être significativement inférieur au nombre de concepts du treillis dont elle est issue. Ainsi, les sous-hiérarchies de Galois sont une alternative au treillis complet lorsque le treillis de Galois est trop grand et que la structure de treillis n'est pas indispensable.

Dans le même ordre d'idées, plusieurs métriques, telles que la *stabilité* [Kuznetsov, 2007] ou la *similarité* [Saquer et Deogun, 2001] ont été proposées afin d'identifier les concepts les plus significatifs selon certains critères. La stabilité d'un concept formel C mesure sa pertinence en mesurant la proportion des sous-ensembles de son extension dont l'intension demeure identique à celle de C. Autrement dit, plus un concept conserve son intension malgré la suppression d'objets, plus il est considéré comme stable. En ce qui concerne la mesure de similarité entre concepts, elle permet, à partir d'un concept donné, d'identifier et de proposer à la navigation des concepts suffisamment proches, qui ne feraient pas partie des parents ou des enfants du concept. Une étude comparative des différentes mesures d'intérêt associées aux concepts formels est présentée dans [Kuznetsov et Makhalova, 2018].

2. Le support d'un concept est le nombre (proportion) d'objets dans son extension.

2.4 Algorithmes

De manière générale, la construction du treillis de concepts à partir d'un contexte formel se complexifie lorsque le contexte devient grand ; d'où l'intérêt de chercher des méthodes permettant de découvrir d'une manière efficace les concepts et de calculer les relations de couverture entre ces derniers. Plusieurs algorithmes ont été proposés dans la littérature pour le calcul des concepts d'un contexte. Étant donné un contexte formel, certains algorithmes permettent uniquement de calculer l'ensemble des concepts tandis que d'autres permettent aussi la construction du treillis associé, i.e., qu'en plus de calculer l'ensemble des concepts, ils calculent également l'ensemble des relations de couverture entre les concepts.

Il existe deux stratégies de base pour construire un treillis de concepts. La distinction majeure entre ces stratégies réside dans la manière d'acquérir les données d'entrée :

- **Les algorithmes batch** [Bordat, 1986 ; Chein, 1969 ; Nourine et Raynaud, 1999] : ils considèrent que les données (contexte formel) sont connues à l'avance. L'évolution des données (ajout d'objets ou d'attributs au contexte) entraîne la reconstruction du treillis dans son entièreté.
- **Les algorithmes incrémentaux** [Godin et al., 1995 ; Valtchev et Missaoui, 2001 ; Van Der Merwe et al., 2004] : ils construisent progressivement le treillis. Ces algorithmes sont apparus pour remédier au problème de la reconstruction du treillis dans le cadre des contextes dynamiques. En effet, suite à une modification du contexte, ces algorithmes effectuent des mises à jour locales du treillis associé.

Comme décrit précédemment, les algorithmes incrémentaux ont l'avantage de s'adapter aux contextes évolutifs et permettent des mises à jour locales du treillis. Ces algorithmes incrémentaux peuvent procéder de deux façons : *incrémentalité par objets* ou *incrémentalité par attributs*.

- L'incrémentalité par objet suppose que le contexte évolue en nombre d'objets (en ligne), donc le treillis est construit par ajout de nouveaux objets qui conduit à une mise à jour structurelle du treillis.
- L'incrémentalité par attributs suppose une évolution du contexte en nombre d'attributs (en colonne), le treillis est alors construit par ajout de nouveaux attributs qui conduit à une mise à jour structurelle du treillis.

Dans le cadre de cette thèse, nous présentons l'algorithme incrémental par objet *AddIntent* [Van Der Merwe et al., 2004] et son algorithme dual *AddExtent* sur lequel se base la procédure `Multi-FCA` de RCA auquel est consacré le chapitre 3.

2.4.1 AddIntent

AddIntent [Van Der Merwe et al., 2004] est un algorithme incrémental qui s'appuie sur un treillis construit à partir des premiers objets du contexte pour intégrer un prochain objet

dans ledit treillis. Comme défini dans d'autres algorithmes [Godin et al., 1995 ; Valtchev et Missaoui, 2001], la construction d'un treillis de concepts peut être décrite en utilisant 4 ensembles de concepts : les concepts *modifiés*, les concepts *générateurs*, les *nouveaux* concepts et les *anciens* concepts. Soient L_1 et L_2 deux treillis de concepts avant et après l'insertion du nouvel objet o, respectivement. Notons o' l'intension de o et (A, B) un concept formel dans L_2 :

- (A, B) est un *nouveau concept* si B n'est l'intension d'aucun concept dans L_1.
- (A, B) est un *concept modifié*, si $B \subseteq o'$ et B est l'intension d'un concept dans L_1.
- Si (A, B) reste *inchangé* de L_1 à L_2, il s'agit d'un *ancien concept*.
- En supposant que (C, D) est un nouveau concept et que (A, B) est un ancien concept, si $B \cap o' = D \neq B$, le concept (A, B) est un générateur[3] du concept (C, D). Sinon, (A, B) est ancien concept.

Soit L un treillis de concepts et o un nouvel objet que l'on souhaite insérer dans L. Le principe de fonctionnement de l'algorithme *AddIntent* peut être résumé comme suit :

1. Déterminer l'intension de l'objet o, notée o'.
2. Identifier dans le treillis L le concept le plus général — c'est-à-dire le plus haut dans la hiérarchie — dont l'intension contient o'. Ce concept est appelé *GeneratorConcept*.
3. Extraire l'ensemble des parents (successeurs directs) de *GeneratorConcept*, noté *GeneratorParents*.
4. Déterminer l'ensemble des parents du nouveau concept à insérer, noté *NewConcept*, en examinant les parents de *GeneratorConcept*. Cet ensemble est appelé *NewParents*.
5. Après traitement des parents de *GeneratorConcept*, le nouveau concept est créé : *NewConcept* ← (*GeneratorConcept.Extent*, o'). Celui-ci est ensuite relié aux concepts de la liste *NewParents*, tout en supprimant les liens existants entre ces derniers et *GeneratorConcept*. Enfin, *NewConcept* est défini comme un voisin supérieur de *GeneratorConcept*.

L'algorithme *AddIntent*, tel que présenté par [Van Der Merwe et al., 2004], ne met pas à jour les extensions des concepts. Cette mise à jour est assurée par la procédure *CreateLatticeWithAddIntent* (algorithme 1, ligne 5), qui construit le treillis de concepts d'un contexte formel (O, A, I) en appliquant successivement *AddIntent* à chaque objet $o \in O$.

2.4.2 *AddExtent* : une version duale de *AddIntent*

Lorsque les données sont amenées à évoluer par l'ajout de nouveaux attributs, il est souvent préférable d'adopter une approche fondée sur les attributs plutôt que sur les objets, afin de faciliter l'intégration de ces nouveaux éléments dans le treillis. Contrairement à l'algorithme *AddIntent*, qui construit le treillis de concepts de manière ascendante en itérant

3. Tout nouveau concept (A, B) possède au moins un générateur. Le (seul) plus général de ces générateurs est appelé le *générateur* canonique de (A, B).

Algorithm 1 CreateLatticeWithAddIntent(O,A, I)

Input : Un contexte formel (O, A, I)

1: *BottomConcept* ← (∅, A)
2: $L \leftarrow \{BottomConcept\}$
3: **for** $o \in O$ **do**
4: *ObjectConcept* = AddIntent(*o', BottomConcept, L*)
5: Ajouter *o* à l'extension de *ObjectConcept* et de tous les concepts au-dessus
6: **end for**

sur les objets, l'algorithme *AddExtent* procède de manière descendante, en parcourant les attributs à partir du concept le plus général (*top-concept*). Ainsi, *AddExtent* constitue la version duale de *AddIntent*.

AddExtent permet d'intégrer un nouvel attribut a dans un treillis. Soient L_1 et L_2 deux treillis de concepts avant et après l'insertion du nouvel attribut a, respectivement. Les 4 ensembles de concepts définis pour *AddIntent* sont également définis pour *AddExtent* avec (A, B) un concept formel dans L_2 et a' l'extension du nouvel attribut a :

- (A, B) est un *nouveau concept* si A n'est l'extension d'aucun concept dans L_1.
- (A, B) est *concept modifié*, si $A \subseteq a'$ et A est l'extension d'un concept dans L_1.
- Si (A, B) reste *inchangé* de L_1 à L_2, il s'agit d'un *ancien concept*.
- En supposant que (C, D) est un nouveau concept et que (A, B) est un ancien concept, si $A \cap a' = C \neq A$, le concept (A, B) est un générateur du concept (C, D). Sinon, (A, B) est ancien concept.

Soit L un treillis de concepts et a un nouvel attribut que l'on souhaite insérer dans L. L'algorithme *AddExtent* fonctionne de manière duale à *AddIntent* et peut se résumer comme suit :

1. Déterminer l'extension de l'attribut a, notée a'.
2. Identifier dans le treillis L le concept le plus spécifique (c'est-à-dire le plus bas dans la hiérarchie) dont l'extension contient a'. Ce concept est désigné par *GeneratorConcept*.
3. Extraire l'ensemble des enfants (prédécesseurs) de *GeneratorConcept*, noté *Generator-Children*.
4. Déterminer l'ensemble des enfants du nouveau concept à insérer (*NewConcept*) en examinant les enfants de *GeneratorConcept*. Cet ensemble est noté *NewChildren*.
5. Créer le nouveau concept – *NewConcept* ← (*extent*, *GeneratorConcept.Intent*) – et le relier aux concepts de la liste *NewChildren*, tout en supprimant les liens existants entre ces derniers et *GeneratorConcept*. Le nouveau concept est alors défini comme voisin inférieur de *GeneratorConcept*.

Tout comme *AddIntent*, *AddExtent* ne met pas à jour les intensions des concepts. La procédure *CreateLatticeWithAddExtent* permet de construire le treillis de concepts d'un

contexte formel (O, A, I) en appliquant successivement *AddExtent* sur chaque attribut, comme présenté à l'algorithme 2.

Algorithm 2 CreateLatticeWithAddExtent(O,A, I)

Input : Un contexte formel (O, A, I)

1: *TopConcept* ← (O, ∅)
2: $L \leftarrow \{TopConcept\}$
3: **for** $a \in A$ **do**
4: *AttributConcept* = AddExtent(*a'*, *TopConcept*, *L*)
5: Ajouter a à l'intension de *AttributConcept* et de tous les concepts en dessous
6: **end for**

2.5 Extensions relationnelles de l'AFC

L'Analyse Formelle des Concepts (AFC) [Ganter et Wille, 1999] a été appliquée à un large éventail de tâches, notamment la recherche d'informations et la découverte de connaissances, ainsi qu'à de nombreux domaines d'application tels que les sciences sociales, l'ingénierie logicielle ou encore la bio-informatique [Ferré et al., 2020 ; Poelmans et al., 2013]. La diversité de ces contextes d'utilisation a rapidement mis en évidence la nécessité d'étendre le cadre classique de l'AFC afin de pouvoir traiter des données plus complexes, structurées ou multi-relationnelles. Cette section introduit d'abord les principales extensions de l'AFC dédiées aux données complexes, avant de présenter un aperçu de celles développées pour les données multi-relationnelles.

Sans prétendre à l'exhaustivité, plusieurs extensions de l'AFC ont été proposées pour traiter des données complexes. Parmi elles, l'Analyse Logique de Concepts (ALC) [Ferré et Ridoux, 2000] et les structures de motifs (*pattern structures*) [Demko et al., 2022 ; Ganter et Kuznetsov, 2001] permettent de manipuler des données où les objets sont décrits par des représentations complexes (par exemple des intervalles numériques, des attributs évalués), plutôt que par de simples ensembles d'attributs binaires. D'autres approches s'intéressent à la prise en compte de l'incertitude dans les descriptions d'objets [Poelmans et al., 2014], comme c'est le cas de l'Analyse Formelle Floue de Concepts (*fuzzy FCA*) [Belohlávek et Vychodil, 2005], où la relation d'incidence entre un objet et un attribut est évaluée par un degré de vérité compris entre 0 et 1, au lieu d'une valeur booléenne stricte. Enfin, les contextes multi-valués [Ganter et Wille, 1999] constituent une extension de l'AFC permettant de traiter des données non binaires où les attributs peuvent prendre plusieurs valeurs, comme dans le cas des variables catégorielles.

D'autres extensions de l'AFC visent à introduire des conditions supplémentaires dans la relation d'incidence entre objets et attributs. L'Analyse Triadique des Concepts (*Triadic Concept Analysis* – TCA) proposée par [Lehmann et Wille, 1995] traite le cas où les don-

nées reposent sur trois ensembles : un ensemble d'objets, un ensemble d'attributs et un ensemble de conditions, liés par une relation ternaire formant un contexte triadique. Autrement dit, elle modélise les situations du type : « un objet o possède un attribut a sous une condition c ». Le résultat de cette analyse est un treillis de concepts triadiques (*concept trilattice*), dont chaque concept est un triplet (A_1, A_2, A_3) où A_1 représente un ensemble d'objets, A_2 un ensemble d'attributs et A_3 un ensemble de conditions. Cette approche a ensuite été généralisée dans [Voutsadakis, 2002] sous le nom d'Analyse Polyadique des Concepts (*Polyadic Concept Analysis*), autorisant un nombre illimité de dimensions pour le contexte.

Pour terminer, plusieurs extensions de l'AFC ont été proposées afin de traiter plus spécifiquement les données multi-relationnelles, pour que les concepts ne dépendent pas uniquement des descriptions individuelles des objets, mais également des structures relationnelles entre des objets interconnectés [Ferré et Cellier, 2020 ; Kötters, 2013 ; Rouane-Hacene et al., 2013 ; Wille, 1997].

Les familles des puissances d'un contexte (FPC) (*power context family*), introduites dans [Wille, 1997] et détaillées, entre autres, dans [Kötters, 2016 ; Prediger et Wille, 1999 ; Wille, 2002], constituent une extension de l'AFC qui vise à exploiter les relations d'arités quelconques dans un jeu de données relationnelles. Elles disposent d'un contexte formel pour chaque arité de relation, c'est-à-dire un contexte d'objets, un contexte de couples d'objets, un contexte de triplets d'objets, etc. Formellement, 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 (comme dans le cas classique de l'AFC, voir définition 2.11 d'un contexte formel).
- $n > 1$, 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 une relation a.

Un treillis de concepts est calculé pour chaque contexte, indépendamment des autres contextes. Les concepts obtenus sont utilisés comme un vocabulaire de types et de relations pour construire des graphes conceptuels similaires à [Chein et Mugnier, 2008 ; Sowa, 1984].

L'Analyse Relationnelle de Concepts (*Relational Concept Analysis* – RCA) [Rouane-Hacene et al., 2013] constitue une extension de l'AFC destinée à l'analyse de données relationnelles compatibles au modèle entité-association [Chen, 1976], c'est-à-dire des données représentées sous la forme de plusieurs tables interconnectées. Les tables décrivant des objets à l'aide d'attributs sont appelées contextes objets-attributs [4], tandis que celles représentant les relations entre objets sont désignées comme contextes objets-objets. Les concepts

4. Il s'agit des contextes formels au sens de l'AFC.

formés dans ce cadre sont qualifiés de relationnels, car ils sont liés à d'autres concepts au moyen d'attributs relationnels, lesquels capturent explicitement les liens inter-objets.

Deux extensions de RCA ont été proposées :

- RCA floue (fuzzy RCA) [Boffa, 2022], qui combine RCA et de fuzzy FCA [Belohlávek et Vychodil, 2005] afin de traiter des ensembles de données multi-relationnelles comportant un certain degré d'imprécision, c'est-à-dire des familles de contextes relationnels flous ;
- l'Analyse Polyadique de Concepts Relationnels (*Polyadic Relational Concept Analysis*) [Bazin et al., 2024], qui associe RCA et l'Analyse Polyadique de Concepts [Voutsadakis, 2002] pour étendre RCA à des ensembles de données relationnelles comportant des relations n-aires.

Graph-FCA (GCA) [Ferré, 2015 ; Ferré et Cellier, 2020] a été proposée comme une extension de l'AFC dans laquelle l'ensemble de données est représenté non plus sous forme tabulaire, mais sous la forme d'un graphe. GCA adopte une approche orientée graphe, où les objets sont modélisés comme des nœuds, les relations comme des arêtes orientées reliant ces nœuds, et les attributs comme des étiquettes portées par les nœuds ou les arêtes. Cette modélisation permet de traiter des relations d'arité quelconque et de calculer des concepts n-aires. Les résultats produits par GCA se présentent sous la forme d'un ensemble de *graph patterns* capturant les structures relationnelles entre les concepts.

Une autre extension de l'AFC aux structures relationnelles est proposée dans [Kötters, 2013], où les treillis de concepts sont directement générés à partir d'une structure relationnelle, celle-ci jouant le rôle de contexte formel dans l'AFC. Comme GCA, cette extension permet de traiter des relations n-aires et de calculer des concepts n-aires, c'est-à-dire des concepts dont les intensions sont équivalentes à des requêtes conjonctives et dont les extensions sont équivalentes aux résultats de ces requêtes, c'est-à-dire des ensembles de n-uplets d'objets.

Enfin, GCA a été fusionnée avec les structures de motifs [Ganter et Kuznetsov, 2001] pour donner naissance à Graph-PS [Ferré, 2023], une extension combinant les avantages des deux paradigmes : la capacité à gérer des descriptions complexes et celle à modéliser explicitement les relations n-aires entre objets.

2.6 Conclusion

Dans ce chapitre, nous avons présenté les notions fondamentales de l'Analyse Formelle de Concepts (AFC), une méthode mathématique d'analyse de données et de découverte de connaissances. L'AFC prend en entrée une table binaire (contexte formel) décrivant des objets par des attributs, et construit un treillis de concepts, où chaque concept correspond à un sous-ensemble d'objets partageant des attributs communs. Nous avons également pro-

posé une synthèse des principales extensions de l'AFC dédiées au traitement de données multi-relationnelles.

Dans le cadre de cette thèse, nous menons une étude comparative de RCA et de GCA qui constituent deux extensions majeures de l'AFC pour l'analyse de données multi-relationnelles. Les chapitres 3 et 4 suivants sont consacrés à une présentation détaillée de RCA et de GCA, respectivement.

Chapitre 3

Analyse Relationnelle de Concepts

Sommaire

3.1 Introduction

Introduite en 2002 dans le cadre de la reconstruction des diagrammes UML [Huchard et al., 2002], l'Analyse Relationnelle de Concepts (*Relational Concept Analysis* - RCA) [Huchard et al., 2007 ; Rouane-Hacene et al., 2013] est une extension de l'Analyse Formelle de Concepts (AFC) pour des données relationnelles. Les concepts ainsi construits sont qualifiés de *relationnels*, dans la mesure où leurs intensions font référence à d'autres concepts.

RCA a démontré son efficacité dans un large éventail d'applications issues de domaines diversifiés. En génie logiciel, le domaine ayant motivé sa création, RCA a été utilisée pour plusieurs tâches notamment la modélisation de la variabilité des familles de produits interconnectées [Carbonnel et al., 2019], la localisation automatisée des fonctionnalités dans les lignes de produits logiciels [Hlad et al., 2021]. On retrouve également des applications de RCA en biologie [Alam et al., 2013], en hydro-écologie [Dolques et al., 2021 ; Nica et al., 2016b, 2016c], en détection de communautés [Guesmi et al., 2016a, 2016b], en développement d'ontologies [Bendaoud et al., 2007 ; Hacene et al., 2008], en prise de décision industrielle [Wajnberg et al., 2019a, 2019b] et plus encore.

Dans la suite de ce chapitre, nous présentons RCA de manière détaillée, en débutant par les notions fondamentales (section 3.2). Une approche naïve d'intégration des relations dans la construction des concepts est ensuite introduite en section 3.3, suivie par la présentation du mécanisme de *scaling relationnel* par lequel RCA intègre les relations dans la description des concepts (section 3.4). Le déroulement global du processus RCA est exposé en section 3.5, accompagné d'une discussion sur l'interprétation des concepts relationnels générés.

3.2 Notions fondamentales de RCA

RCA est une extension de l'AFC pour le traitement des données relationnelles, c'est-à-dire des données représentées par plusieurs tables et les associations qui les relient. Les tables décrivant des objets au moyen d'attributs sont appelées contextes *objets-attributs* (contextes formels), tandis que celles qui encodent les relations entre objets sont appelées contextes *objets-objets* (contextes relationnels). Les concepts formés sont dits *relationnels*, car les intensions qu'ils renferment établissent des liens avec d'autres concepts. Nous commençons par définir le modèle de données utilisé en RCA appelé *Famille Relationnelle de Contextes (FRC)*.

3.2.1 Famille Relationnelle de Contextes (FRC)

Les données en RCA sont décrites par un ensemble de contextes formels noté **K** et un ensemble de contextes relationnels[1] noté **R** représentant les relations d'incidence entre ensembles d'objets de **K**. Ainsi, une relation $r \subseteq O_{i_1} \times O_{i_2}$ donne lieu à un contexte relationnel dont les lignes et les colonnes correspondent aux objets de O_{i_1} et O_{i_2} respectivement.

Définition 3.1 (Famille Relationnelle de Contextes (FRC)). Une FRC est une paire $(\mathbf{K}, \mathbf{R})$ où :

- **K** = $\{\mathcal{K}_i\}_{i=1,\ldots,n}$ est un ensemble de contextes objets-attributs $\mathcal{K}_i = (O_i, A_i, I_i)$ décrivant des objets par leurs attributs.
- **R** = $\{r_k\}_{k=1,\ldots,m}$ est un ensemble de contextes objets-objets r_k encodant les relations entre les objets où $r_k \subseteq O_{i_1} \times O_{i_2}$ avec $i_1, i_2 \in \{1, \ldots, n\}$.

Pour des raisons pratiques, une relation $r \subseteq O_{i_1} \times O_{i_2}$ est traitée sous la forme d'une fonction définie par $r : O_{i_1} \to \mathcal{P}(O_{i_2})$ [Rouane-Hacene et al., 2013]. Pour les mêmes raisons, quelques fonctions auxiliaires sont introduites pour soutenir le raisonnement centré sur les relations.

Définition 3.2 (Les fonctions *dom(r)* et *codom(r)* des relations). Soit (**K**, **R**) une FRC. Une paire de fonctions établit une correspondance entre les relations de **R** et les ensembles d'objets de l'ensemble **O** = $\{O_i \,|\, \mathcal{K}_i = (O_i, A_i, I_i) \in \mathbf{K}\}$ des objets de la FRC.

- La fonction de domaine $dom : \mathbf{R} \to \mathbf{O}$ où $dom(r) = O_{i_1}$ si et seulement si $\forall (x, y) \in r$, $x \in O_{i_1}$.
- La fonction de codomaine $codom : \mathbf{R} \to \mathbf{O}$ où $codom(r) = O_{i_2}$ si et seulement si $\forall (x, y) \in r, y \in O_{i_2}$.

Une autre fonction permet de regrouper des relations par rapport à leur domaine, c'est la fonction de contexte que l'on note rel.

Définition 3.3 (La fonction *rel(K)* des contextes). L'ensemble des relations issues d'un contexte $\mathcal{K}_i = (O_i, A_i, I_i)$ donné est défini par la fonction $rel : \mathbf{K} \to \mathcal{P}(\mathbf{R})$ qui pour chaque contexte associe l'ensemble de relations issu de celui-ci : $rel(\mathcal{K}_i) = \{r \in \mathbf{R} \,|\, dom(r) = O_i\}$.

3.2.2 Exemple de Famille Relationnelle de Contextes

Pour illustrer la notion de Famille Relationnelle de Contextes (FRC) définie précédemment, ajoutons au contexte formel $\mathcal{K}_{Person}$ du tableau 2.1, les contextes formels $\mathcal{K}_{Garage}$ et $\mathcal{K}_{Car}$

1. Les termes *contextes objets-attributs* et *contextes formels* sont utilisés de façon interchangeable. Il en va de même pour les termes *contextes objets-objets* et *contextes relationnels*.

Tableau 3.1 – FRC_1 (**K**, **R**) avec **K** = $\{\mathcal{K}_{Person}, \mathcal{K}_{Car}, \mathcal{K}_{Garage}\}$ et **R** = $\{$*owner, sell, maintain*$\}$.

$\mathcal{K}_{Person}$	male	female	city	countryside	single	married
Alice		×	×		×	
Bob	×			×		×
Charlie	×		×		×	
Julie		×		×		×

owner	Alice	Bob	Charlie	Julie
car1		×		
car2				×
car3			×	
car4	×			
car5				×
car6		×		

$\mathcal{K}_{Car}$	Renault	Peugeot	Tesla	family	sport	sedan
car1	×			×		
car2		×		×		
car3			×		×	
car4			×			×
car5		×			×	
car6	×					×

sell	car1	car2	car3	car4	car5	car6
A	×					×
B			×	×		
C		×			×	
D						

$\mathcal{K}_{Garage}$	manufacturer	chain	service
A	×		
B		×	
C	×		
D			×

maintain	car1	car2	car3	car4	car5	car6
A	×					×
B			×	×		
C						
D		×			×	

qui décrivent les garages et les voitures par leurs caractéristiques respectivement. Considérons la relation *owner* entre les voitures et les personnes, les relations *sell* et *maintain* entre les garages et les voitures. L'ensemble de ces tables présenté dans le tableau 3.1 forme la FRC_1 (**K**, **R**) avec **K** = $\{\mathcal{K}_{Person}, \mathcal{K}_{Car}, \mathcal{K}_{Garage}\}$ et **R** = $\{$*owner, sell, maintain*$\}$.

La relation *owner* indique pour chaque voiture son propriétaire. À titre d'exemple, l'incidence *owner*$(car1, Bob)$ signifie que la voiture $car1$ a pour propriétaire Bob. La relation *sell* encode quant à elle la vente des voitures par les garages. Ainsi, *sell*$(A, car1)$ indique que le $garage\ A$ vend la voiture $car1$. Pour terminer, la relation *maintain* renseigne la maintenance des voitures par des garages. Notamment, *maintain*$(D, car2)$ signifie que le $garage\ D$ fait la maintenance de la voiture $car2$. Il en résulte les éléments suivants :

- $dom($*owner*$) = O_{Car}$ et $codom($*owner*$) = O_{Person}$,
- $dom($*sell*$) = dom($*maintain*$) = O_{Garage}$ et $codom($*sell*$) = codom($*maintain*$) = O_{Car}$.
- $rel(\mathcal{K}_{Garage}) = \{$*sell, maintain*$\}$, $rel(\mathcal{K}_{Car}) = \{$*owner*$\}$ et $rel(\mathcal{K}_{Person}) = \varnothing$ [2]

Une notation des relations qui indiquent directement leurs domaines et codomaines est *nom_relation*$(dom, codom)$. Par exemple, on obtient *owner*(O_{Car}, O_{Person}), *sell*(O_{Garage}, O_{Car}) et *maintain*(O_{Garage}, O_{Car}).

À ce stade, nous savons comment construire les treillis de chacun des contextes for-

2. Ceci signifie que les entités O_{Person} du contexte $\mathcal{K}_{Person}$ ne constituent le domaine d'aucune relation et ne sont par conséquent pas impactées par l'ensemble des relations.

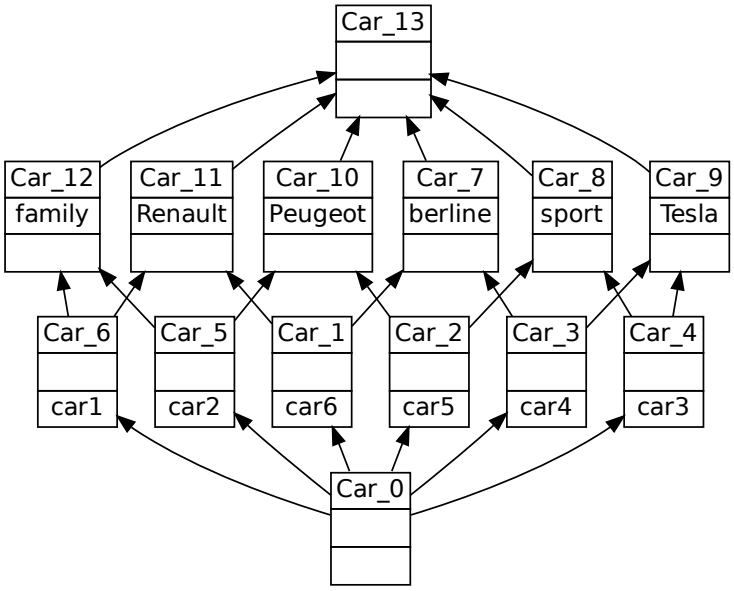

FIGURE 3.1 – Treillis $\mathcal{L}^0_{Car}$ du contexte $\mathcal{K}_{Car}$.

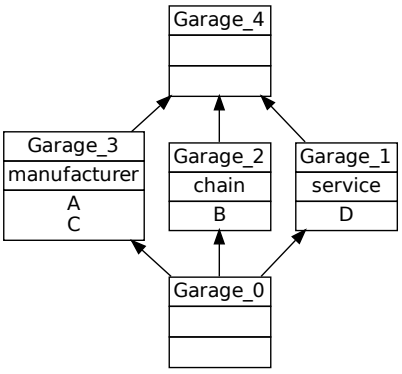

FIGURE 3.2 – Treillis $\mathcal{L}^0_{Garage}$ du contexte $\mathcal{K}_{Garage}$.

mels $\mathcal{K}_{Person}$, $\mathcal{K}_{Car}$ et $\mathcal{K}_{Garage}$. Les figures 3.1 et 3.2 présentent respectivement les treillis des contextes $\mathcal{K}_{Car}$ et $\mathcal{K}_{Garage}$. Quant au treillis du contexte $\mathcal{K}_{Person}$, il a déjà été présenté à la figure 2.3.

La grande problématique tient à l'intégration des relations inter-objets dans le processus d'analyse. Comment intégrer dans les concepts de $\mathcal{K}_{Car}$ le fait que les voitures ont des propriétaires ? C'est-à-dire, le fait que les voitures sont associées à des personnes à travers la relation *owner*. De même, comment intégrer dans les concepts de $\mathcal{K}_{Garage}$ l'information selon laquelle les garages vendent et entretiennent des voitures ? En clair, comment intégrer le fait que les garages sont reliés à des personnes via les relations *sell* et *maintain* ? En effet, pour tout contexte donné $\mathcal{K}_i = (O_i, A_i, I_i)$, *chaque relation* r *issue de* $\mathcal{K}_i$ *(*$\forall\, r \in rel(\mathcal{K}_i)$*), est utile à la description des éléments* $o \in O_i$. La section suivante présente une approche naïve pour l'intégration des relations dans la description des concepts.

3.3 Approche naïve de scaling des relations

Une manière simple et quelque peu naïve de représenter les liens entre objets consisterait à assimiler les liens à des attributs standards à valeur unique [ROUANE-HACENE et al., 2013]. En d'autres termes, pour chaque $\mathcal{K}_i = (O_i, A_i, I_i) \in \mathbf{K}$, on étend A_i avec des attributs $a_{r:\bar{o}}$ correspondant aux couples constitués d'une relation $r \in rel(\mathcal{K}_i)$ (voir définition 3.3), et d'un objet $\bar{o}$ tel que $(o, \bar{o}) \in r$ pour un certain $o \in O_i$.

Pour la FRC_1 présentée dans le tableau 3.1, rappelons que $rel(\mathcal{K}_{Garage}) = \{$*sell, maintain*$\}$,

$rel(\mathcal{K}_{Car}) = \{owner\}$ et $rel(\mathcal{K}_{Person}) = \varnothing$. Le scaling des relations de FRC_1 par l'approche naïve va consister d'une part en l'extension du contexte $\mathcal{K}_{Car}$ grâce à la relation *owner* par les attributs de la forme *owner* $:person$ où $person \in codom(owner) = O_{Person}$. L'intersection de la ligne $car_i \in O_{Car}$ avec la colonne *owner* $:person$ contient une croix si et seulement si $(car_i, person) \in owner$. Le tableau 3.2 illustre le contexte étendu $\mathcal{K}_{Car+}$, obtenu à partir de l'extension naïve du contexte initial $\mathcal{K}_{Car}$. D'autre part, l'extension du contexte $\mathcal{K}_{Garage}$ se fait par l'ajout des attributs de la forme *sell* $:car_i$ et *maintain* $:car_i$ formés sur les relations *sell*(O_{Garage}, O_{Car}) et *maintain*(O_{Garage}, O_{Car}) respectivement. Le tableau 3.3 illustre le contexte $\mathcal{K}_{Garage+}$, correspondant à l'extension naïve du contexte initial $\mathcal{K}_{Garage}$. Quant au contexte $\mathcal{K}_{Person}$, il reste inchangé (et son treillis en conséquence), car ses entités ne constituent le domaine d'aucune relation.

Tableau 3.2 – Contexte $\mathcal{K}_{Car+}$ correspondant à l'extension naïve du contexte initial $\mathcal{K}_{Car}$.

$\mathcal{K}_{Car+}$	Renault	Peugeot	Tesla	family	sport	berline	owner :Alice	owner :Bob	owner :Charlie	owner :Julie
car1	×			×				×		
car2		×		×						×
car3			×		×				×	
car4			×			×	×			
car5		×			×					×
car6	×					×		×		

Tableau 3.3 – Contexte $\mathcal{K}_{Garage+}$ correspondant à l'extension naïve du contexte $\mathcal{K}_{Garage}$.

$\mathcal{K}_{Garage+}$	manufacturer	chain	service	sell :car1	sell :car2	sell :car3	sell :car4	sell :car5	sell :car6	maintain :car1	maintain :car2	maintain :car3	maintain :car4	maintain :car5	maintain :car6
A	×			×					×	×					×
B		×				×	×					×	×		
C	×				×			×							
D			×								×			×	

Le treillis dérivé du contexte $\mathcal{K}_{Car+}$ (tableau 3.2) est représenté dans la figure 3.3. En comparaison au treillis de la figure 3.1 obtenu sur le contexte initial $\mathcal{K}_{Car}$, nous obtenons les mêmes concepts (14 concepts au total) à la différence que les concepts de $\mathcal{K}_{Car+}$ renseignent également l'information sur les propriétaires des voitures. Par exemple, le concept

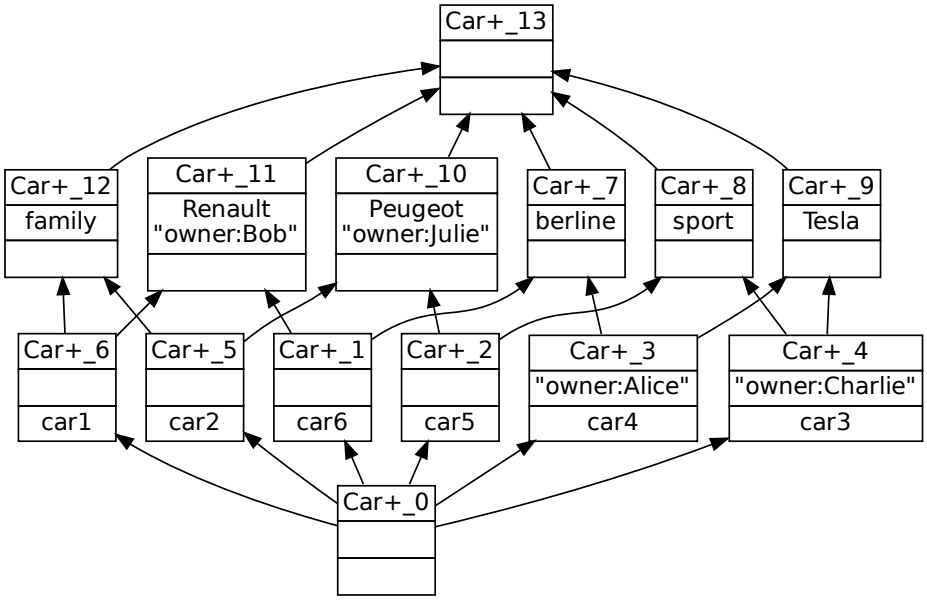

FIGURE 3.3 – Treillis du contexte $\mathcal{K}_{Car+}$ (tableau 3.2).

Car+_11 (figure 3.3) indique que les voitures $\{car1, car6\}$ sont de marque *Renault et ont pour propriétaire* Bob, contrairement au concept *Car_11* (figure 3.1) qui donne uniquement l'information que les voitures $\{car1, car6\}$ sont de marque *Renault*.

De même, la figure 3.4 présente le treillis du contexte $\mathcal{K}_{Garage+}$ avec 7 concepts contre 5 concepts pour le treillis du contexte initial $\mathcal{K}_{Garage}$ (figure 3.2). Comme nous pouvons le voir sur ces deux figures, le treillis de la figure 3.4 apporte plus de précision à la description des garages à savoir *qu'ils vendent et maintiennent des voitures*. Pour illustration, les concepts **Garage+_2** et **Garage+_4** qui ne sont pas produits avec $\mathcal{K}_{Garage}$, indiquent que le $garage\, A$ vend et maintient les voitures $\{car1, car6\}$ et le $garage\, C$ vend les voitures $\{car2, car5\}$. Nous pouvons aussi remarquer la différence d'informations entre les intensions du concept *Garage+_1* et son équivalent *Garage_1* (figure 3.2), de même que pour le concept *Garage+_3* et son concept équivalent *Garage_2* (figure 3.2).

Cette approche naïve de mise à échelle des relations a tendance à beaucoup augmenter la taille des contextes formels en nombre d'attributs. L'ensemble d'attributs A_i d'un contexte $\mathcal{K}_i = (O_i, A_i, I_i)$ se voit augmenter de la taille du codomaine de chaque relation $r \in rel(\mathcal{K}_i)$; ce qui peut déboucher sur des treillis de grande taille avec des concepts dont les intensions sont difficiles à interpréter, car beaucoup d'éléments doivent être pris en compte pour la description d'un concept. De plus, cette approche ne garantit pas l'obtention des concepts pertinents dans la mesure où elle n'aide pas à capturer les relations entre des groupes d'objets (concepts). Pour illustration, $\{car1, car6\}$ représente l'extension du concept *Car+_11*, une abstraction consisterait à indiquer que les garages du concept *Garage+_4* (à savoir $garage\, A$) vendent et maintiennent les voitures du concept *Car+_11*, ce

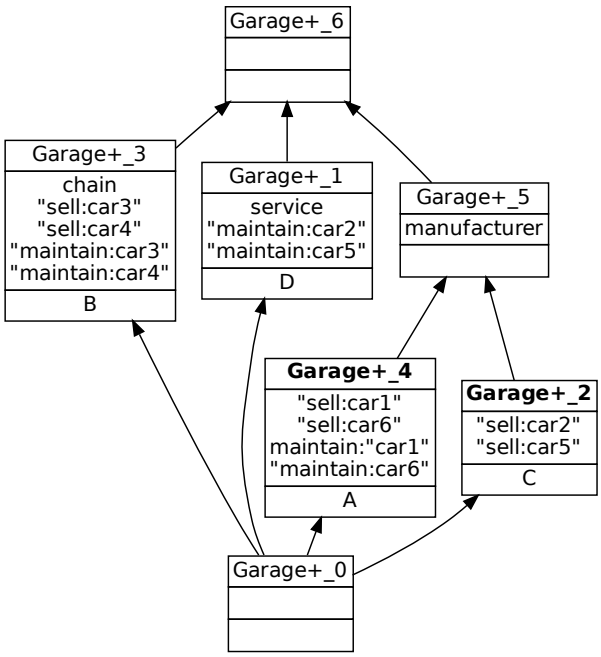

Figure 3.4 – Treillis du contexte $\mathcal{K}_{Garage+}$ (tableau 3.3).

qui pourrait être indiqué grâce à deux attributs au lieu de quatre comme c'est le cas dans l'intension de *Garage+_4*. Il en est de même pour le $garage\,C$ (*Garage+_2*) qui vend les voitures de *Car+_10* ou du $garage\,D$ (Garage+_1) qui maintient les voitures de *Car+_10*. Ce type d'abstraction consistant à représenter et quantifier les relations entre les concepts, faciliterait l'interprétation des intensions des concepts et permettrait d'avoir des descriptions de haut niveau et plus pertinentes des concepts.

Dans la section suivante, nous présentons le *scaling relationnel*, qui est le mécanisme par lequel RCA intègre les relations inter-objets dans la construction des concepts de sorte à pouvoir les abstraire en des relations inter-concepts.

3.4 Scaling relationnel

La question de savoir comment intégrer les relations inter-objets au cours du processus de RCA admet une variété de réponses qui dépendent des objectifs de l'analyse conceptuelle. Pour traiter les données relationnelles, RCA travaille en enrichissant les contextes objets-attributs avec de nouveaux attributs, appelés *attributs relationnels* inspirés de la restriction des rôles en Logique de Descriptions (LD) [Baader et al., 2003]. La force des liens capturés dépend du quantificateur d'échelle, également appelé opérateur de *scaling*.

3.4.1 Opérateurs de scaling

La mise à l'échelle d'un contexte objets-attributs se fait par ajout *d'attributs relationnels* qui mettent en exergue les relations entre concepts. Ces attributs relationnels sont obtenus en associant des quantificateurs d'échelle (opérateurs de scaling), des relations et des concepts.

Définition 3.4 (Attribut relationnel). Un attribut relationnel est une expression $qr(C)$, où q est un quantificateur d'échelle, r une relation (nom de la relation) et C un concept dont l'extension contient des objets de $codom(r)$.

Selon les besoins de l'application, différents quantificateurs peuvent être choisis pour définir l'importance du lien entre les objets d'un ensemble et ceux d'un autre ensemble. Nous introduisons ici les deux quantificateurs couramment utilisés : *le quantificateur existentiel* $\exists$ *et le quantificateur universel strict* $\exists\forall$. Il existe d'autres quantificateurs qui sont en majorité des variantes de ces derniers [Braud et al., 2018 ; Rouane-Hacene et al., 2013].

Dans les définitions suivantes, nous considérons deux contextes formels $\mathcal{K} = (O, A, I)$ et $\mathcal{K}_r = (O_r, A_r, I_r)$, et un contexte relationnel r avec $dom(r) = O$ et $codom(r) = O_r$; $\mathcal{C}_r$ l'ensemble des concepts produits sur $\mathcal{K}_r$. Pour $o \in O$, l'image de o par la relation r est représentée par $r(o) = \{o_i \in O_r \mid (o, o_i) \in r\}$. Pour illustration, dans la FRC du tableau 3.1, prenons $\mathcal{K}$ pour le contexte $\mathcal{K}_{Garage}$, $\mathcal{K}_r$ pour le contexte $\mathcal{K}_{Car}$ et r pour la relation *sell* avec $dom(r) = O_{Garage}$ (ensemble des garages) et $codom(r) = O_{Car}$ (ensemble des voitures).

Définition 3.5 (Scaling existentiel). Pour tout objet $o \in O$ et tout concept $C_i \in \mathcal{C}_r$, si $r(o) \cap Extent(C_i) \neq \varnothing$, alors l'attribut relationnel $\exists r(C_i)$ est ajouté à l'ensemble d'attributs de o.

Le scaling existentiel est qualifié de *large*, dans la mesure où il n'impose pas une relation fortement contrainte entre l'image $r(o)$ d'un objet $o \in O$ et l'extension du concept C_i associé à l'attribut relationnel $\exists r(C_i)$. Ainsi, nous pouvons classer les garages selon qu'ils vendent au moins une voiture d'un certain groupe (concept).

Dans la figure 3.1, $Ext^3(Car\_12) = \{car1, car2\}$. Donc, l'attribut relationnel $\exists$ *sell(Car_12)* est ajouté aux garages qui vendent au moins l'une des voitures $\{car1, car2\}$. Par exemple, *sell*$(garage\ A) = \{car1, car6\}$ donc, *sell*$(garage\ A) \cap Ext(Car\_12) \neq \varnothing$. Par conséquent, $\exists$ *sell(Car_12)* devient un attribut de $garage\ A$. Le treillis du contexte $\mathcal{K}_{Car}$ compte 14 concepts, ce qui conduit à 14 attributs relationnels de la forme $\exists$ *sell*(Car_i), où Car_i est un concept du treillis $\mathcal{L}^0_{Car}$ du contexte $\mathcal{K}_{Car}$.

Le tableau 3.4 illustre le résultat de l'échelonnage existentiel du contexte $\mathcal{K}_{Garage}$ basé sur la relation *sell* et les concepts $\mathcal{C}_{car}$ du treillis $\mathcal{L}^0_{Car}$ des voitures (figure 3.1). Remarquons que l'attribut $\exists$*sell*(Car_0) (en rouge) n'est partagé par aucun objet garage car,

3. Ext est mis pour $Extent$, la fonction d'extension.

Tableau 3.4 – Scaling existentiel du contexte initial $\mathcal{K}_{Garage}$ des garages en fonction de la relation *sell* et les concepts du treillis $\mathcal{L}^0_{Car}$ des voitures (figure 3.1).

$\mathcal{K}_{Garage}$	manufacturer	chain	service	∃ sell(Car_13)	∃s sell(Car_11)	∃ sell(Car_0)	∃ sell(Car_10)	∃ sell(Car_9)	∃ sell(Car_6)	∃ sell(Car_5)	∃ sell(Car_12)	∃ sell(Car_2)	∃ sell(Car_4)	∃ sell(Car_8)	∃ sell(Car_1)	∃ sell(Car_3)	∃ sell(Car_7)
A	×			×	×				×		×				×		×
B		×		×				×					×	×		×	×
C	×			×			×			×	×	×		×			
D			×														

$Ext(Car\_0) = \varnothing$. Cet attribut sera par conséquent un attribut du concept *Garage_0* (concept $\perp$ des garages, figure 3.5), dont l'extension est également vide.

La figure 3.5 rend compte du treillis de concepts résultant du scaling existentiel du contexte $\mathcal{K}_{Garage}$ tel que présenté dans le tableau 3.4. Ce treillis contient 10 concepts contre 5 concepts pour le treillis $\mathcal{L}^0_{Garage}$ (figure 3.2). Soit au total 5 concepts supplémentaires qui émergent avec le scaling existentiel : les concepts dont les identifiants sont marqués en gras, par exemple **Garage_5** (figure 3.5). En plus de ces nouveaux concepts, les intensions de certains concepts (du contexte initial) sont enrichies par les informations relationnelles. Pour illustration, le concept *Garage_3* d'extension $\{A, C\}$ (figure 3.5), contrairement à ses concepts équivalents *Garage+_5* (figure 3.4) et *Garage_3* (figure 3.2), contient dans son intension l'attribut $\exists sell(Car\_12)$ qui signifie que chaque garage (de son extension) vend au moins une voiture de Car_12. Plus précisément, $Ext(Car\_12) = \{car1, car2\}$ et nous avons les correspondances $sell(A, car1)$ et $sell(C, car2)$.

Comme nous venons de le voir, le scaling existentiel est un encodage très large, dans la mesure où il suffit de l'existence d'une relation entre un objet et les objets d'un autre concept pour l'ajout de l'attribut relationnel formé audit objet. Ce qui, en pratique, peut entraîner des liens relativement faibles entre les concepts. Nous poursuivons avec le *scaling universel strict*, qui impose une contrainte plus forte entre les concepts.

Définition 3.6 (Scaling universel strict). Pour tout objet $o \in O$ et tout concept $C_i \in \mathcal{C}_r$, si $r(o) \neq \varnothing$ et $r(o) \subseteq Extent(C_i)$, alors l'attribut relationnel $\exists\forall r(C_i)$ est ajouté à l'ensemble d'attributs de o.

Le scaling universel strict est qualifié de *restreint* ou *étroit* en comparaison au scaling existentiel, qualifié de large. En effet, la contrainte d'inclusion imposée entre l'image $r(o)$ d'un objet $o \in O$ et l'extension du concept C_i associé à l'attribut relationnel $\exists\forall r(C_i)$ est plus forte que celle appliquée dans le cas du scaling existentiel.

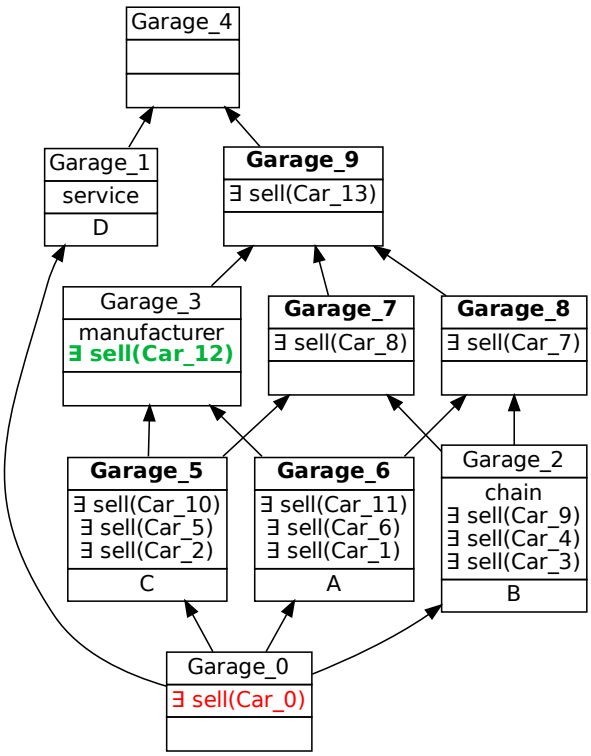

FIGURE 3.5 – Treillis du contexte $\mathcal{K}_{Garage}$ après scaling existentiel (tableau 3.4).

Par exemple, le concept Car_11 de la figure 3.1 a pour extension $\{car1, car6\}$ et on a $\textit{sell}(garage\,A) = \{car1, car6\} \subseteq Ext(Car\_11)$, par conséquent, l'attribut relationnel $\exists\forall$ $\textit{sell}(Car\_11)$ est ajouté aux attributs de l'objet $garage\,A$.

Dans la même logique, avec le concept Car_13 qui est le top ($\top$) concept des voitures (regroupe toutes les voitures), l'attribut $\exists\forall \textit{sell}(Car\_13)$ est ajouté comme attribut à tous les garages ayant vendu une voiture (il s'agit des garages $\{A, B, C\}$, qui constituent l'extension du concept **Garage_7** dans la figure 3.6), car l'ensemble des voitures vendus par chaque garage est inclus dans $Ext(Car\_13)$.

Le tableau 3.5 est le résultat du scaling universel strict sur le contexte $\mathcal{K}_{Garage}$ en fonction de la relation *sell* et les concepts du treillis $\mathcal{L}^0_{Car}$ des voitures (figure 3.1). Dans ce tableau, nous pouvons remarquer que la relation d'incidence entre les garages et les attributs relationnels est très peu dense, en comparaison avec le tableau 3.4. Au total 10 attributs relationnels sur 14 (à partir de $\exists\forall \textit{sell}(Car\_0)$, en rouge, tableau 3.5), ne sont partagés par aucun objet et vont par conséquent constituer l'intension du concept $Garage\_0$ ($\bot$ concept des garages, figure 3.6), car son extension est également vide.

La figure 3.6 met en évidence le treillis associé au contexte du tableau 3.5. Ce treillis compte 8 concepts contre 5 concepts pour le treillis du contexte initial (figure 3.2) ; les 3 concepts supplémentaires sont identifiés en gras (par exemple **Garage_7**). En comparaison

TABLEAU 3.5 – Scaling universel strict du contexte $\mathcal{K}_{Garage}$ en fonction de la relation *sell* et les concepts du treillis $\mathcal{L}^0_{Car}$ des voitures (figure 3.1).

$\mathcal{K}_{Garage}$	manufacturer	chain	service	∃∀ sell(Car_13)	∃∀ sell(Car_11)	∃∀ sell(Car_10)	∃∀ sell(Car_9)	∃∀ sell(Car_0)	∃∀ sell(Car_6)	∃∀ sell(Car_5)	∃∀ sell(Car_12)	∃∀ sell(Car_2)	∃∀ sell(Car_4)	∃∀ sell(Car_8)	∃∀ sell(Car_1)	∃∀ sell(Car_3)	∃∀ sell(Car_7)
A	×			×	×												
B		×		×			×										
C	×			×		×											
D			×														

TABLEAU 3.6 – Principaux opérateurs de scaling implémentés dans les outils de RCA

Opérateur	Notation	Attribut relationnel	Contraintes
Existentiel	$\exists$	$\exists r(C)$	$r(o) \cap Ext(C) \neq \varnothing$
Universel strict	$\exists\forall$	$\forall r(C)$	$r(o) \neq \varnothing$ et $r(o) \subseteq Ext(C)$
Universal-percent	$\exists\forall_{\geq n\%}$	$\exists\forall_{\geq n\%} r(C)$	$\lvert r(o) \cap Ext(C)\rvert \geq n\lvert r(o)\rvert/100$ et $\lvert r(o) \cap Ext(C)\rvert > 0$
Contains strict	$\exists \supseteq$	$\exists \supseteq r(C)$	$Ext(C) \neq \varnothing$ et $Ext(C) \subseteq r(o)$
Contains-Percent	$\exists \supseteq_{\geq n\%}$	$\exists \supseteq_{\geq n\%} r(C)$	$\lvert r(o) \cap Ext(C)\rvert \geq n\lvert Ext(C)\rvert/100$ et $\lvert r(o) \cap Ext(C)\rvert > 0$

avec le treillis obtenu par le scaling existentiel (figure 3.5), on obtient 2 concepts en moins avec le scaling universel strict ce qui illustre bien son caractère restreint par rapport au scaling existentiel.

Outre les quantificateurs existentiel $\exists$ et universel strict $\exists\forall$ de scaling, d'autres variantes de quantificateurs peuvent être utilisées pour préciser la force des liens entre les ensemble d'objets, notamment les quantificateurs : *universal-percent* $\exists\forall_{\geq n\%}$, *contains strict* $\exists \supseteq$ *et contains-percent* $\exists \supseteq_{\geq n\%}$. Soient une relation r, un objet $o \in dom(r)$ et C un concept tel que $Ext(C) \subseteq codom(r)$, le tableau 3.6 récapitule les principaux quantificateurs intégrés dans les outils qui implémentent RCA à l'instar de RCAExplore [DOLQUES et al., 2019] et FCA4J [GUTIERREZ et al., 2022].

Un *ordre de généralité* peut être défini sur les attributs relationnels et sur les quantificateurs de scaling.

Définition 3.7 (Relation de généralité $\preceq_A$ sur les attributs relationnels). Soient deux attributs relationnels $a_1 = q_1 r_1(C_1)$ et $a2 = q_2 r_2(C_2)$ tels que $r_1 \subseteq O_3 \times O_1$, $r_2 \subseteq O_3 \times O_2$, $Ext(C_1) \subseteq O_1$ et $Ext(C_2) \subseteq O_2$: a_1 est plus général que a_2 (noté $a_1 \preceq_A a_2$) si et seulement

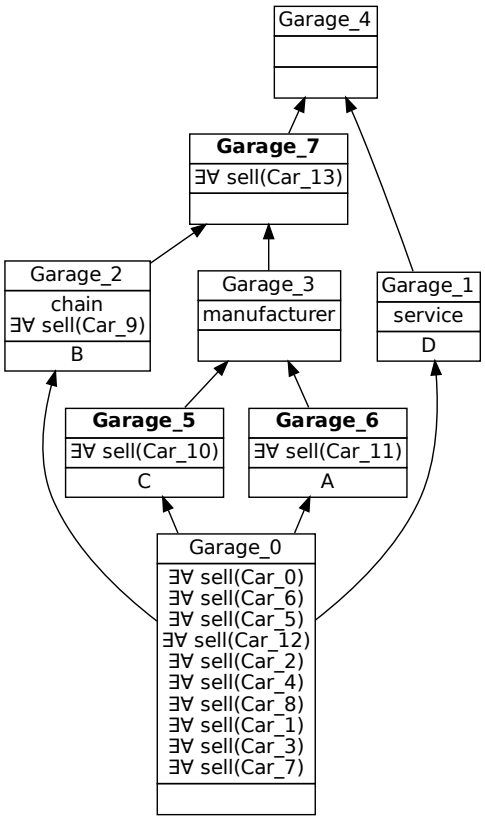

FIGURE 3.6 – Treillis du contexte $\mathcal{K}_{Garage}$ après scaling universel strict (tableau 3.5).

si $\{a_2\}' \subseteq \{a_1\}'$ [BRAUD et al., 2018].

Définition 3.8 (Relation de généralité $\preceq_S$ sur les quantificateurs). Soient q_1 et q_2 deux quantificateurs de scaling ; q_1 est plus général que q_2 ($q_1 \preceq_S q_2$) si $\forall r,\ \forall C,\ q_1 r(C) \preceq_A q_2 r(C)$ [BRAUD et al., 2018].

À titre d'exemple, la relation $\exists \preceq_S \exists\forall$ signifie que l'extension du concept introduisant $\exists r(C)$ inclut celle du concept introduisant $\exists\forall r(C)$. La figure 3.7 illustre cette hiérarchie de généralité entre les quantificateurs présentés dans le tableau 3.6, allant des plus spécifiques vers les plus généraux, du haut vers le bas (par exemple, $\exists\forall$ est plus spécifique que $\exists\forall_{\geq 60\%}$). Cette propriété permet d'établir une relation entre les treillis obtenus à partir de deux quantificateurs q_1 et q_2 tels que $q_1 \preceq_S q_2$. En particulier, il existe une projection des concepts construits avec q_2 vers ceux construits avec q_1. Autrement dit, pour chaque concept C_1 du treillis construit avec q_2, il existe un concept C_2 du treillis construit avec q_1 tel que $Ext(C_1) \subseteq Ext(C_2)$. Par exemple, le treillis de concepts de la figure 3.6 construit avec le quantificateur $\exists\forall$ se projette dans celui de la figure 3.5, obtenu avec le quantificateur $\exists$. Des précisions supplémentaires sur ces formalismes et leurs démonstrations figurent dans [BRAUD et al., 2018].

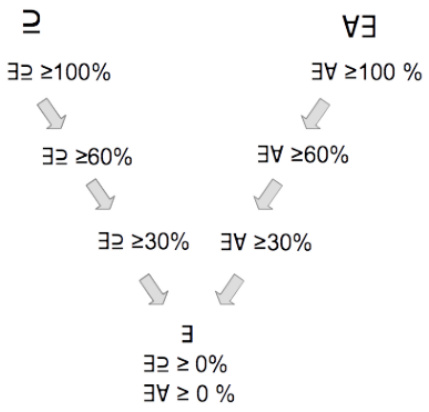

Figure 3.7 – Relation de généralité sur les quantificateurs – de [Braud et al., 2018].

La section suivante présente le processus d'extension d'un contexte objets-attributs par intégration des relations inter-objets via les attributs relationnels.

3.4.2 Extension relationnelle d'un contexte

L'ajout d'attributs relationnels à un contexte objets-attributs s'effectue via une opération de mise à l'échelle, basée sur un quantificateur. Afin de fournir une définition générique de cette opération, introduisons une fonction générique κ [Braud et al., 2018] qui fait correspondre un quantificateur d'échelle q, une relation r et un sous-ensemble d'objets du codomaine de r à un sous-ensemble d'objets du domaine de r.

$$\kappa : Q \times \mathbf{R} \times \bigcup_{i=1,\ldots,n} 2^{O_i} \rightarrow \bigcup_{j=1,\ldots,n} 2^{O_j} \; avec$$

- $Q = \{\exists, \exists\forall, \exists \ni, \ldots\}$: un ensemble de quantificateurs d'échelle
- $\mathbf{R}$: un ensemble de relations (contextes objets-objets)
- O_i : le codomaine d'une relation $r \in \mathbf{R}$
- O_j : le domaine d'une relation $r \in \mathbf{R}$

κ renvoie le groupe d'objets du domaine qui sont connectés pour r et q avec un certain groupe d'objets du codomaine. Plus précisément, pour r, q et un concept C sur le codomaine de r, $\kappa(r, q, Ext(C))$ renvoie l'ensemble des objets qui possèdent l'attribut relationnel $qr(C)$. Par exemple, $Ext(Car\_12) = \{car1, car2\}$ (figure 3.1) et $\kappa(\exists, \textit{sell}, Ext(Car\_12)) = \{A, C\}$; donc $\exists \textit{sell}(Car\_12)$ est un attribut commun aux garages A et C (voir tableau 3.4 ou treillis de la figure 3.5).

Définition 3.9 (Extension relationnelle partielle). Considérons les contextes objets-attributs $\mathcal{K}_k = (O_k, A_k, I_k)$ et $\mathcal{K}_l = (O_l, A_l, I_l)$, la relation $r \subseteq O_k \times O_l$, le treillis $\mathcal{L}_l$ du

contexte $\mathcal{K}_l$ et q un quantificateur d'échelle. L'opérateur de scaling $\mathbb{S}_{(r,q),\mathcal{L}_l}$ sur $\mathcal{K}_k$ donne l'extension relationnelle partielle $\mathbb{S}_{(r,q),\mathcal{L}_l}(\mathcal{K}_k) = (O^+, A^+, I^+)$ avec :

- $O^+ = O_k$,
- $A^+ = \{qr(C) \,|\, C \in \mathcal{L}_l\}$,
- $I^+ = \bigcup_{C \in \mathcal{L}_l} \kappa(q, r, Ext(C)) \times \{qr(C)\}$.

L'extension relationnelle d'un contexte objets-attributs $\mathcal{K}_k$ est obtenue en le joignant avec toutes les extensions relationnelles partielles générées à partir des relations qui ont O_k comme domaine et leur opérateur d'échelle assigné. Ainsi, considérons une famille relationnelle de contextes qui ne contient que le contexte $\mathcal{K}_k$ ci-dessus mentionné, le contexte $\mathcal{K}_l$ et la relation r avec $dom(r) = O_k$ et $codom(r) = O_l$. L'extension relationnelle de $\mathcal{K}_k$ lors de l'affectation du quantificateur q à r est l'union du contexte initial avec son extension relationnelle partielle comme formulée dans l'équation 3.1.

$$\mathcal{K}_k \cup \mathbb{S}_{(r,q),\mathcal{L}_l}(\mathcal{K}_k) = (O_k, A_k \cup A^+, I_k \cup I^+) \tag{3.1}$$

L'équation 3.1 correspond en effet à l'*apposition* des contextes $\mathcal{K}_k$ et $\mathbb{S}_{(r,q),\mathcal{L}_l}(\mathcal{K}_k)$. L'opérateur d'apposition sur deux contextes $\mathcal{K}_1 = (O_1, A_1, I_1)$ et $\mathcal{K}_2 = (O_2, A_2, I_2)$, tels que $O_1 = O_2 = O$ est noté $\mathcal{K}_1 \,|\, \mathcal{K}_2 = (O, A_1 \cup A_2, I_1 \cup I_2)$ et représente le contexte sur ces objets où l'ensemble d'attributs et l'incidence sont obtenus par l'union des composantes respectives de $\mathcal{K}_1$ et $\mathcal{K}_2$ [Ganter et Wille, 1999].

À titre d'illustration, le tableau 3.7 représente $\mathbb{S}_{(sell,\exists),\mathcal{L}_{\mathcal{K}_{Car}}(\mathcal{K}_{Garage})}$ [4], l'extension partielle de $\mathcal{K}_{Garage}$ sur $r = sell(O_{Garage}, O_{Car})$, $q = \exists$ et les concepts du treillis de $\mathcal{L}^0_{Car}$ (figure 3.1). Ainsi, l'extension relationnelle de $\mathcal{K}_{Garage}$ est illustrée dans le tableau 3.4 et correspond à l'union du contexte initial $\mathcal{K}_{Garage}$ (voir FRC_1 du tableau 3.1) et l'extension partielle présentée dans le tableau 3.7.

La section suivante présente l'extension relationnelle complète d'un contexte par rapport à l'ensemble des relations dont il est la source.

3.4.3 Extension relationnelle complète d'un contexte

Dans RCA, un contexte $\mathcal{K}_k$ est mis à l'échelle en utilisant toutes les relations issues de $\mathcal{K}_k$, c'est-à-dire les relations de l'ensemble $rel(\mathcal{K}_k)$. Pour exprimer formellement le contexte obtenu en augmentant $\mathcal{K}_k$ avec tous les attributs relationnels résultants, ce que nous appelons *extension relationnelle complète* de $\mathcal{K}_k$, il faut prendre en compte les treillis des contextes associés aux codomaines des relations de $rel(\mathcal{K}_k)$.

4. L'extension se fait sur $\mathcal{L}_{\mathcal{K}_{Car}}$, car $\mathcal{K}_{Car}$ est le contexte associé au codomaine de la relation *sell*.

Tableau 3.7 – Extension relationnelle partielle du contexte $\mathcal{K}_{Garage}$ en fonction de la relation *sell*, du quantificateur $\exists$ et des concepts du treillis $\mathcal{L}^0_{Car}$ des voitures (figure 3.1) .

$\mathbb{S}_{(sell,\exists),\mathcal{L}_{\mathcal{K}_{Car}}(\mathcal{K}_{Garage})}$	∃ sell(Car_13)	∃s sell(Car_11)	∃ sell(Car_0)	∃ sell(Car_10)	∃ sell(Car_9)	∃ sell(Car_6)	∃ sell(Car_5)	∃ sell(Car_12)	∃ sell(Car_2)	∃ sell(Car_4)	∃ sell(Car_8)	∃ sell(Car_1)	∃ sell(Car_3)	∃ sell(Car_7)
A	×	×				×		×				×		×
B	×				×					×	×		×	×
C	×			×			×	×	×		×			
D														

Désignons par **L** l'ensemble des treillis correspondant aux contextes de **K**. Soit $rel(\mathcal{K}_k) = \{r_l\}_{l=1,\dots,m_k}$, et pour chaque r_l, soit $\mathcal{L}_{il} \in \mathbf{L}$ le treillis sur $O_{il} = codom(r_l)$. Introduisons la fonction $\rho : \mathbf{R} \to Q$ qui associe à chaque relation objets-objets $r_l \in \mathbf{R}$ un quantificateur d'échelle $q \in Q = \{\exists, \exists\forall, \dots\}$. Sous ces hypothèses, *l'extension relationnelle complète* de $\mathcal{K}_k$ par rapport à ρ et **L**, notée $\mathbb{E}_{\rho,\mathbf{L}}(\mathcal{K}_k)$, est définie comme l'apposition de $\mathcal{K}_k$ avec les résultats respectifs de sa mise à l'échelle sur chacune des relations $r_l \in rel(\mathcal{K}_k)$. Autrement dit, l'extension relationnelle complète du contexte $\mathcal{K}_k$ consiste à appliquer successivement le *scaling* associé à chaque $r_l \in rel(\mathcal{K}_k)$, selon le quantificateur défini par ρ.

Définition 3.10 (Extension relationnelle complète d'un contexte). Étant donné $(\mathbf{K}, \mathbf{R})$ une FRC, avec un ensemble de treillis **L**, ρ un constructeur de scaling et un contexte $\mathcal{K}_k \in \mathbf{K}$ avec $rel(\mathcal{K}_k) = \{r_l\}_{l=1,\dots,m_k}$ l'extension relationnelle complète du contexte $\mathcal{K}_k$ par rapport à ρ et **L** est :

$$\mathbb{E}_{\rho,\mathbf{L}}(\mathcal{K}_k) = \mathcal{K}_k \,|\, \mathbb{S}_{(r_1,\rho(r_1)),\mathcal{L}_{i1}}(\mathcal{K}_k) \,|\, \dots \,|\, \mathbb{S}_{(r_{m_k},\rho(r_{m_k})),\mathcal{L}_{im_k}}(\mathcal{K}_k)$$

Comme exemple, rappelons que $rel(\mathcal{K}_{Garage}) = \{sell,\ maintain\}$ avec O_{Garage} et O_{Car} pour domaine et codomaine des deux relations respectivement. Pour la FRC_1 du tableau 3.1, notons $\mathbf{L} = \{\mathcal{L}_{Garage}, \mathcal{L}_{Car}, \mathcal{L}_{Person}\}$. L'extension relationnelle complète $\mathbb{E}_{\rho,\mathbf{L}}(\mathcal{K}_{Garage})$ de $\mathcal{K}_{Garage}$, avec $\rho(sell) = \exists$ et $\rho(maintain) = \exists$, va consister en son extension relationnelle sur les relations *sell* (comme présentée dans le tableau 3.7) et *maintain* en correspondance. En d'autres termes, on a :

$$\mathbb{E}_{\rho,\mathbf{L}}(\mathcal{K}_{Garage}) = \mathcal{K}_{Garage} \,|\, \mathbb{S}_{(maintain,\exists),\mathcal{L}_{Car}}(\mathcal{K}_{Garage}) \,|\, \mathbb{S}_{(sell,\exists),\mathcal{L}_{Car}}(\mathcal{K}_{Garage}) \tag{3.2}$$

Le tableau 3.8 récapitule $\mathbb{E}_{\rho,\mathbf{L}}(\mathcal{K}_{Garage})$, l'extension relationnelle complète de $\mathcal{K}_{Garage}$ en mettant en évidence l'extrait de l'extension partielle correspondant à chacune des relations *maintain* et *sell*.

TABLEAU 3.8 – Extension relationnelle complète du contexte $\mathcal{K}_{Garage}$ en fonction des relations $\{maintain, sell\}$, du quantificateur $\exists$ et des concepts du treillis $\mathcal{L}^0_{Car}$ (figure 3.1) .

$\mathbb{E}_{\rho,\mathbf{L}}(\mathcal{K}_{Garage})$	$\mathcal{K}_{Garage}$			$\mathbb{S}_{(maintain,\exists),\mathcal{L}_{Car}}(\mathcal{K}_{Garage})$				$\mathbb{S}_{(sell,\exists),\mathcal{L}_{Car}}(\mathcal{K}_{Garage})$				
	manufacturer	chain	service	$\exists$ maintain(Car_13)	$\exists$ maintain(Car_12)	...	$\exists$ maintain(Car_1)	$\exists$ sell(Car_13)	$\exists$ sell(Car_12)	$\exists$ sell(Car_11)	...	$\exists$ sell(Car_1)
A	×			×	×	...	×	×	×	×	...	×
B		×		×		...		×			...	
C	×					...		×	×		...	
D			×	×	×	...					...	

Lorsque l'extension relationnelle complète pour chaque contexte d'une FRC est calculée, on parle de *l'extension relationnelle complète* de cette FRC.

3.4.4 Extension relationnelle complète d'une FRC

L'opérateur $\mathbb{E}_{\rho,\mathbf{L}}(\mathcal{K}_k)$ d'extension relationnelle complète d'un contexte ci-dessus peut être utilisé pour couvrir l'ensemble des contextes d'une FRC. L'idée étant d'obtenir l'ensemble des extensions complètes pour chaque contexte de la FRC. Comme dans la section précédente, l'extension se fait par rapport aux paramètres ρ et $\mathbf{L}$ telle que formulée à la définition 3.11 suivante.

Définition 3.11 (Extension relationnelle complète d'une FRC). Étant donnée $(\mathbf{K}, \mathbf{R})$ une FRC ayant pour ensemble de contextes $\mathbf{K} = \{\mathcal{K}_1, \ldots, \mathcal{K}_n\}$ et pour ensemble de treillis $\mathbf{L}$, ainsi que le constructeur de scaling ρ, l'extension relationnelle complète de $\mathbf{K}$ est composée de toutes les extensions relationnelles complètes de tous les $\mathcal{K}_k \in \mathbf{K}$:

$$\mathbb{E}^*_{\rho,\mathbf{L}}(\mathbf{K}) = \{\mathbb{E}_{\rho,\mathbf{L}}(\mathcal{K}_1), \ldots, \mathbb{E}_{\rho,\mathbf{L}}(\mathcal{K}_n)\}$$

L'application de $\mathbb{E}^*$ à un ensemble de contextes donne un ensemble de contextes étendus où chaque contexte (objets-attributs) individuel est une extension par attributs de son homologue dans l'ensemble initial des contextes. Il est clair qu'à chacun des contextes étendus (augmentés), correspond un treillis potentiellement plus grand.

À titre d'exemple, l'application de $\mathbb{E}^*$ avec $\rho(sell) = \rho(maintain) = \rho(owner) = \exists$ sur la FRC_1 (tableau 3.1) se constitue des extensions relationnelles complètes de ses contextes $\mathcal{K}_{Garage}$, $\mathcal{K}_{Car}$ et $\mathcal{K}_{Person}$ respectivement :

- L'extension $\mathbb{E}_{\rho,\mathbf{L}}(\mathcal{K}_{Garage})$ a été présentée dans le tableau 3.8.
- Quant à l'extension relationnelle de $\mathcal{K}_{Car}$, on a $\mathbb{E}_{\rho,\mathbf{L}}(\mathcal{K}_{Car}) = \mathcal{K}_{Car} \mid \mathbb{S}_{(owner,\exists),\mathcal{L}_{Person}}(\mathcal{K}_{Car})$ qui correspond au tableau 3.9.
- Pour ce qui est de $\mathcal{K}_{Person}$, on a tout simplement $\mathbb{E}_{\rho,\mathbf{L}}(\mathcal{K}_{Person}) = \mathcal{K}_{Person}$, car il n'existe aucune relation issue de $\mathcal{K}_{Person}$.

Tableau 3.9 – Extension relationnelle complète du contexte $\mathcal{K}_{Car}$ en fonction de la relation *owner*, du quantificateur $\exists$ et des concepts du treillis $\mathcal{L}^0_{Car}$ des voitures (figure 3.1) .

$\mathbb{E}^{\exists,\mathbf{L}}(\mathcal{K}_{Car})$	Renault	Peugeot	Tesla	family	sport	berline	∃ owner(Person_9)	∃ owner(Person_6)	∃ owner(Person_0)	∃ owner(Person_7)	∃ owner(Person_4)	∃ owner(Person_2)	∃ owner(Person_8)	∃ owner(Person_1)	∃ owner(Person_3)	∃ owner(Person_5)
car1	×			×			×	×							×	×
car2		×		×			×			×				×		×
car3			×		×		×	×				×	×			
car4			×			×	×			×	×		×			
car5		×			×		×			×				×		×
car6	×					×	×	×							×	×

Comme mentionné précédemment, l'extension relationnelle complète d'une FRC (à partir des treillis de l'ensemble initial des contextes formels) conduit à des contextes avec plus d'attributs et par conséquent à des treillis potentiellement plus grands. Ainsi, la question de savoir si ces nouveaux concepts devraient être réutilisés dans une nouvelle étape de mise à l'échelle se pose. En d'autres termes, faut-il répéter l'étape de mise à l'échelle, cette fois avec des informations conceptuelles plus complètes à utiliser comme base de mise à l'échelle ? Dans la section suivante, nous présentons le processus RCA qui intègre de manière itérative les informations relationnelles dans chaque contexte.

3.5 Déroulement de RCA

Le processus de RCA suit une logique itérative. L'étape d'initialisation consiste à construire le treillis de concepts pour chaque contexte formel de départ, puis, chaque itération comprend deux étapes : (1) la mise à l'échelle relationnelle des contextes formels et (2) la construction des treillis de concepts pour chaque contexte formel étendu. L'itération suivante est basée sur les résultats de l'itération précédente en appliquant l'AFC aux contextes

formels étendus, et le processus se termine lorsque les treillis de deux étapes consécutives sont équivalents. Ce processus est formalisé par la méthode `Multi-FCA`, présentée dans la section suivante.

3.5.1 Méthode Multi-FCA

Le processus de RCA est décrit en détail par l'algorithme 3 qui reprend la procédure `Multi-FCA` définie dans [Rouane-Hacene et al., 2013]. Cette méthode Multi-FCA fonctionne selon une logique itérative : chaque fois que les contextes formels de la FRC sont étendus, leurs treillis correspondants s'étendent également. Cette méthode représente un schéma de calcul plutôt qu'un algorithme précis, car de nombreux choix algorithmiques sont laissés à l'analyste. À titre d'exemple, avec la primitive *BUILD-LATTICE* pour la construction d'un ordre sur les concepts, différents algorithmes tels que *Ares* [Dicky et al., 1995] (pour calculer un sous-ordre du treillis) ou *Iceberg* [Stumme et al., 2002] (pour calculer un sup-demi-treillis) peuvent être utilisés.

Algorithm 3 Processus ARC

1: **Proc** `Multi-FCA`
2: **Input** : $(\mathbf{K}, \mathbf{R}) = (\{\mathcal{K}_i\}_{i=1,\ldots,n}, \mathbf{R})$ une FRC, ρ un constructeur de scaling
3: **Output** : $\mathbf{L}$ un tableau [1,..,n] de treillis
$\triangleright$ ρ la est fonction qui à chaque relation associe un quantificateur d'échelle
4: $p \leftarrow 0\,;\ halt \leftarrow$ **false**
5: **for** i from 1 to n **do**
6: $\mathcal{K}_i^0 \leftarrow SCALE(\mathcal{K}_i)$ $\triangleright$ Dans le cas où $\mathcal{K}_i$ est multi-valué
7: $\mathbf{L}^0[i] \leftarrow BUILD-LATTICE(\mathcal{K}_i^0)$
8: **end for**
9: **while** not halt **do**
10: $p = p + 1$
11: **for** i from 1 to n **do**
12: $\mathcal{K}_i^p \leftarrow EXTEND-CONTEXT(\mathcal{K}_i^{p-1}, \rho, \mathbf{L}^{p-1})$
13: $\mathbf{L}^p[i] \leftarrow UPDATE-LATTICE(\mathcal{K}_i^p, \mathbf{L}^{p-1}[i])$
14: **end for**
15: $halt \leftarrow \bigwedge_{i=1,\ldots,n} ISOMORPHIC(\mathbf{L}^p[i], \mathbf{L}^{p-1)}[i])$
16: **end while**

La procédure `Multi-FCA` (algorithme 3) se déroule en plusieurs étapes et se décrit comme suit.

1) Initialisation (lignes 5 à 8). Chaque contexte $\mathcal{K}_i^0$ est obtenu à partir de $\mathcal{K}_i$ en appliquant une mise à l'échelle conceptuelle aux attributs multi-valués de $\mathcal{K}_i$ à l'aide de la primitive

$SCALE$ (ligne 6). Le treillis de concepts associé à $\mathcal{K}_i^0$ est ensuite construit (ligne 7) en utilisant la primitive $BUILD - LATTICE$ et est stocké dans $\mathbf{L}^0[i]$. À la fin de l'initialisation, la variable $\mathbf{L}^0$ contient l'ensemble de treillis de tous les contextes $\mathcal{K}_i^0$.

2) Étape $p > 0$ (lignes 9 à 14). À l'étape p, pour chaque relation $r_k \subseteq O_i \times O_j$, les concepts du treillis $\mathbf{L}^{p-1}[j]$ du codomaine (O_j) de r_k sont utilisés pour étendre le contexte $\mathcal{K}_i^{p-1}$ de son domaine (O_i) en utilisant la primitive $EXTEND - CONTEXT$ (ligne 12). On obtient un contexte étendu $\mathcal{K}_i^p$. Ensuite, le treillis $L^p[i]$ correspondant est obtenu en utilisant la primitive $UPDATE - LATTICE$ (ligne 13). Pour les deux primitives de construction et de mise à jour du treillis, le choix des algorithmes est libre.

3) Arrêt du processus (ligne 15). Le processus s'arrête lorsque pour tous les contextes objets-attributs, les treillis correspondants sont isomorphes sur deux étapes consécutives. Autrement dit, le processus s'arrête lorsque toutes les paires de treillis $\mathbf{L}^p[i]$ et $\mathbf{L}^{p+1}[i]$ pour un certain p sont équivalents.

Graphiquement, la méthode Multi-FCA du processus RCA peut se résumer par l'organigramme de la figure 3.8 ci-dessous.

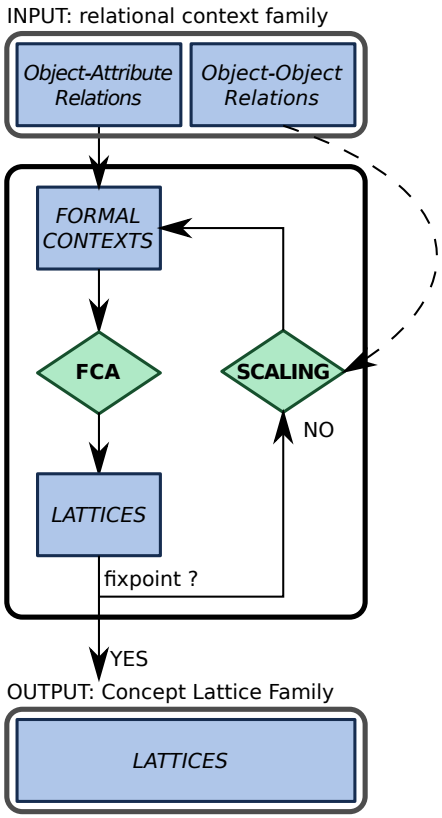

Figure 3.8 – Représentation schématique du processus RCA (@X. Dolques).

À titre d'illustration, appliquons la méthode Multi-FCA à la FRC_1 (tableau 3.1) composée des contextes $\mathbf{K} = \{\mathcal{K}_{Garage}, \mathcal{K}_{Person}, \mathcal{K}_{Car}\}$ et des relations $\mathbf{R} = \{sell, maintain, owner\}$.

L'initialisation ($p = 0$) consiste en la construction des treillis pour chacun des contextes de $\mathcal{K}_{Garage}$ ($\mathcal{L}^0_{Garage}$), $\mathcal{K}_{Person}$ ($\mathcal{L}^0_{Person}$) et $\mathcal{K}_{Car}$ ($\mathcal{L}^0_{Car}$) présentés dans les figures 3.2, 2.3 et 3.1, respectivement. Utilisons le quantificateur $\exists$ pour la mise à échelle de toutes les relations. Pour rappel, aucune relation n'est issue du contexte $\mathcal{K}_{Person}$ c'est-à-dire, que O_{Person} n'est le domaine d'aucune relation, par conséquent, le contexte $\mathcal{K}_{Person}$ (ainsi que le treillis) reste fixe le long du processus RCA.

À **l'étape** $p = 1$ (step 1 : première extension relationnelle complète de la FRC), les treillis construits à l'étape $p = 0$ sont utilisés pour étendre les contextes et mettre à jour les treillis. Les concepts de $\mathcal{L}^0_{Person}$ sont utilisés en association avec la relation *owner* pour étendre $\mathcal{K}_{Car}$ en $\mathcal{K}^1_{Car}$, dont le contexte étendu résultant a été illustré dans le tableau 3.9.
De même, les concepts de $\mathcal{L}^0_{Car}$ sont utilisés en association avec les relations *sell* et *maintain* pour étendre le contexte $\mathcal{K}_{Garage}$ (en $\mathcal{K}^1_{Garage}$) comme formulé à l'équation (3.2) et présenté dans le tableau 3.8. À cette étape, la condition d'arrêt n'est pas vérifiée et on passe à l'étape $p = 2$.

À **l'étape** $p = 2$, les treillis de l'étape précédente sont utilisés pour étendre les contextes. Comme le contexte $\mathcal{K}_{Person}$ reste invariable le long des itérations de scaling ($\mathcal{K}^2_{Person} = \mathcal{K}^1_{Person} = \mathcal{K}^0_{Person}$), on s'attend à avoir $\mathcal{K}^1_{Car} = \mathcal{K}^2_{Car}$ et $\mathcal{L}^1_{Car} = \mathcal{L}^2_{Car}$ en conséquence. Pour ce qui est du contexte $\mathcal{K}_{Garage}$, il est étendu avec le treillis $\mathcal{L}^1_{Car} \neq \mathcal{L}^0_{Car}$ et on a :

$$\mathbb{E}_{\rho,\mathbf{L}}(\mathcal{K}_{Garage}) = \mathcal{K}^2_{Garage} = \mathcal{K}_{Garage} \,|\, \mathbb{S}_{(sell,\exists),\mathcal{L}^1_{Car}}(\mathcal{K}_{Garage}) \,|\, \mathbb{S}_{(maintain,\exists),\mathcal{L}^1_{Car}}(\mathcal{K}_{Garage})$$

À la fin de cette itération, on obtient que $\mathcal{L}^1_{Car} = \mathcal{L}^2_{Car}$, $\mathcal{L}^1_{Garage} = \mathcal{L}^2_{Garage}$ et bien évidement, $\mathcal{L}^1_{Person} = \mathcal{L}^2_{Person}$ ce qui marque l'arrêt du processus RCA. La figure 3.9 présente le treillis $\mathcal{L}^2_{Car}$ à la fin du processus RCA. Ce treillis contient 3 concepts supplémentaires (**Car_14**, **Car_15**, **Car_16**) par rapport au treillis initial $\mathcal{L}^0_{Car}$.

De même, la figure 3.10 illustre le treillis $\mathcal{L}^2_{Garage}$ à la fin du processus RCA. Ce treillis compte 13 concepts contre 5 concepts par rapport au treillis initial $\mathcal{L}^0_{Garage}$. Les 8 concepts supplémentaires sont marqués en gras, par exemple le concept **Garage_7**. Par ailleurs, il est à noter que $\mathcal{L}^1_{Garage} = \mathcal{L}^2_{Garage}$, mais que les intensions de certains concepts de $\mathcal{L}^2_{Garage}$ ont été mises à jour par rapport à leurs intensions dans $\mathcal{L}^1_{Garage}$. Ces mises à jour résultent de l'intégration des attributs relationnels induits par les concepts **Car_14**, **Car_15** et **Car_16** du treillis $\mathcal{L}^1_{Car}$, comme illustré en vert dans la figure 3.10.

Pour résumer, le tableau 3.10 récapitule le nombre de concepts pour chaque contexte au fil des itérations du processus RCA.

Nous expliquons comment interpréter une intension d'un concept relationnel dans la section suivante.

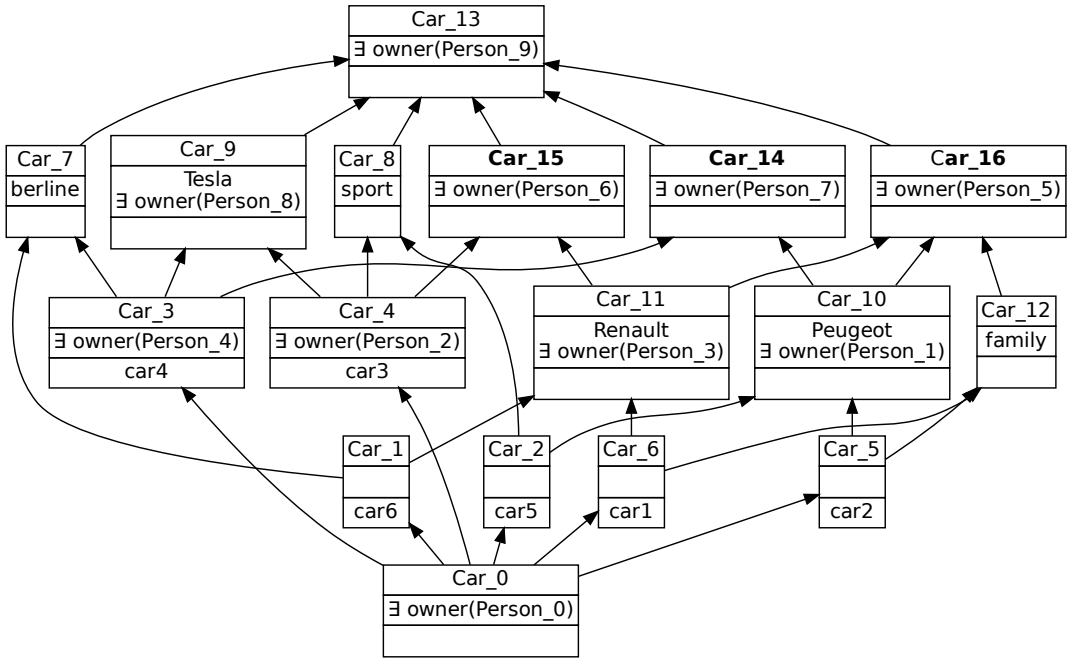

Figure 3.9 – Treillis des voitures $\mathcal{L}^2_{Car}$ (étape 2 de RCA).

Tableau 3.10 – Nombre de concepts pour chaque itération de scaling.

	$p = 0$	$p = 1$	$p = 2$
$\mathcal{K}^p_{Person}$	10	10	10
$\mathcal{K}^p_{Car}$	14	17	17
$\mathcal{K}^p_{Garage}$	5	13	13

3.5.2 Interprétation des concepts relationnels

De nouvelles abstractions qui émergent des treillis relationnels caractérisent les liens inter-objets (comme nous l'avons expliqué à la section 3.4.2). Pour clarifier, le concept **Car_14** du treillis $\mathcal{L}^2_{Car}$ (figure 3.9) introduit par l'attribut relationnel $\exists$*owner*$(Person\_7)$ n'appartient pas au treillis initial $\mathcal{L}^0_{Car}$ (figure 3.1). Il s'agit d'un nouveau concept relationnel dont les objets, à savoir $\{car2, car4, car5\}$ sont décrits de manière purement relationnelle, indiquant que ce sont des voitures dont les propriétaires sont des femmes, car le concept $Person\_7$ a pour extension $\{Julie, Alice\}$ et pour intension $\{$*female*$\}$.

En outre, certaines intensions de concepts des contextes de départ sont complétées par une partie relationnelle, ce qui permet d'affiner la description des objets concernés. Ainsi, le concept $Garage\_3$ du treillis initial $\mathcal{L}^0_{Garage}$ (figure 3.2) représente la classe des garages de type *manufacturer*. Dans le treillis final des garages $\mathcal{L}^2_{Garage}$ (figure 3.10), l'intension du même concept ($Garage\_3$), enrichie par les attributs relationnels $\exists$*sell*(Car_12)

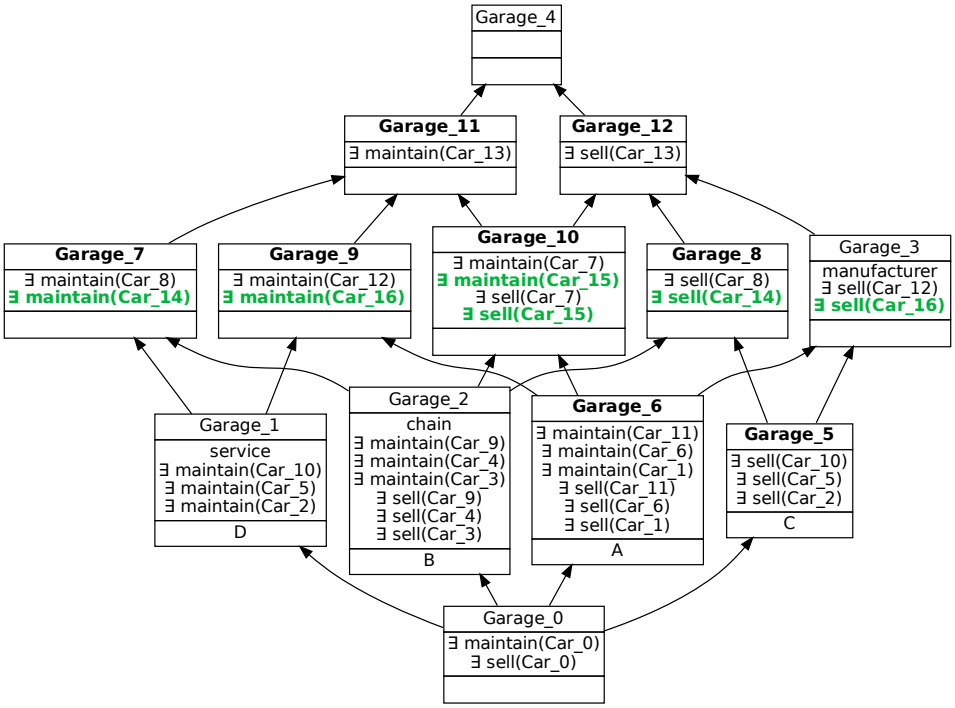

FIGURE 3.10 – Treillis des garages $\mathcal{L}^2_{Garage}$ (étape 2 de RCA).

et $\exists sell(Car\_16)$, indique qu'il s'agit des garages qui vendent des voitures de type *family* (Car_12) et des voitures (Car_16) dont les propriétaires (concept $Person\_5$) ont les caractéristiques *married* et *contryside*. Plus précisément, le concept $Person\_5$ (figure 2.3) a pour extension $\{Bob, Julie\}$ et pour intension $\{$*married, countryside*$\}$. En résumé, les treillis relationnels de la FRC_1 fournis par RCA relient les concepts des garages aux concepts des voitures qui sont à leur tour reliés aux concepts des personnes, comme l'illustre l'extrait de la figure 3.11.

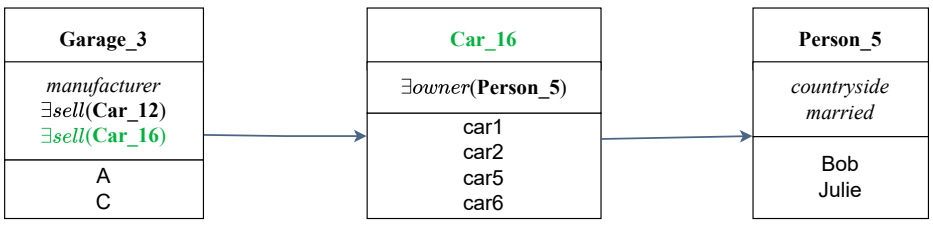

FIGURE 3.11 – Extrait du résultat de RCA reliant les garages de type *manufacturer* aux personnes de caractéristiques *married* et *contryside*.

3.5.3 Itérations dans Multi-FCA

La procédure `Multi-FCA` présentée à la section 3.5.1 calcule les attributs relationnels et étend les contextes au fil des itérations. En fait, dans de nombreux cas, *il n'est pas nécessaire d'itérer* : un ordre approprié des tâches d'analyse pour les contextes individuels devrait permettre de l'éviter [Rouane-Hacene et al., 2013].

On peut s'imaginer une structuration de la FRC sous forme d'un graphe dans lequel les sommets représentent les contextes objets-attributs, et les arêtes représentent les contextes objets-objets. Pour illustration, la figure 3.12 présente la structure graphique de la FRC_1 (tableau 3.1) . À condition que ce graphe soit un graphe acyclique orienté (*Directed Acyclic Graph* - DAG), un tri topologique des contextes fournirait un ordre total compatible avec les dépendances induites par les relations entre les contextes. Ainsi, un ordre toplogique sur le graphe de la figure 3.12 donne la succession des contextes (sommets) dans l'ordre $\mathcal{K}_{Garage}$, $\mathcal{K}_{Car}$, $\mathcal{K}_{Person}$ où chaque contexte $\mathcal{K}_i$ apparaît bien avant ses successeurs, c'est-à-dire les contextes de codomaines des relations issues de $\mathcal{K}_i$.

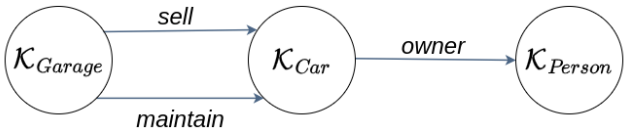

Figure 3.12 – Structure graphique de la FRC_1 (tableau 3.1).

En conséquence, l'analyse des contextes selon *l'ordre topologique inverse* garantit qu'un contexte $\mathcal{K}_i$ n'est traité que lorsque tous les treillis nécessaires à la mise à l'échelle des relations dans $rel(\mathcal{K}_i)$ ont déjà été construits jusqu'à leur forme de point fixe. Dans l'exemple de la FRC_1, le calcul des treillis devrait s'effectuer suivant l'ordre $\mathcal{K}_{Person}, \mathcal{K}_{Car}, \mathcal{K}_{Garage}$ ce qui se traduit par : (1) construction du treillis de $\mathcal{K}_{Person}$, (2) extension relationnelle du contexte $\mathcal{K}_{Car}$ à l'aide des concepts de $\mathcal{K}_{Person}$, puis construction du treillis associé, (3) extension relationnelle de $\mathcal{K}_{Garage}$ à l'aide des concepts de $\mathcal{K}_{Car}$ (étendu), puis construction du treillis associé. Suivant cet ordre de traitement des contextes, nous n'avons pas besoin de plusieurs itérations de scaling sur cet exemple pour calculer l'ensemble de treillis relationnels associé.

Cependant, afin de couvrir tout le spectre des FRC possibles, y compris les FRC cycliques, l'expression plus générale de la méthode `Multi-FCA` est celle liée à un point fixe comme présenté dans l'algorithme 3.

3.6 Conclusion

Nous avons présenté dans ce chapitre l'Analyse Relationnelle de Concepts (*Relational Concept Analysis* - RCA) qui est une extension de l'Analyse Formelle de Concepts (AFC) aux

données multi-relationnelles et dont l'objectif est de capturer les relations inter-objets dans la construction des concepts. À partir d'une famille relationnelle de contextes, RCA produit un ensemble de treillis inter-connectés par des attributs relationnels reliant des concepts entre eux. RCA dispose de divers quantificateurs d'échelle ($\exists, \exists\forall, \supseteq$, etc.) permettant de quantifier les relations entre objets et concepts, ce qui permet une analyse flexible.

RCA se limite au traitement de relations binaires (relations reliant deux catégories d'objets) et nécessite des transformations et modélisations supplémentaires pour le traitement des relations d'arité $n > 2$. Dans le chapitre 4, nous présentons Graph-FCA, la deuxième extension de l'AFC étudié dans cette thèse. Graph-FCA étend l'AFC aux graphes et a pour spécificité la prise en compte des relations $n(\geq 2)$-aires et le calcul des concepts $n(\geq 1)$-aires (concepts des relations).

Chapitre 4

Analyse Conceptuelle de Graphes (Graph-FCA)

Sommaire

4.1 Introduction

Graph-FCA (GCA) [Ferré, 2015 ; Ferré et Cellier, 2020] étend l'Analyse Formelle de Concepts (AFC) aux graphes et vise à calculer des structures conceptuelles dans les graphes, de la même manière que l'AFC découvre les structures conceptuelles dans les données tabulaires (contextes formels). En GCA, les nœuds du graphe jouent le rôle d'objets dans l'AFC, les attributs de l'AFC sont les labels (attributs unaires) des nœuds et les attributs n(>1)-aires sont les labels des arêtes connectant n nœuds. Ainsi, GCA se caractérise par sa capacité à prendre en compte des relations de toute arité, tout en permettant le calcul de concepts n-aires. En termes d'applications, GCA a été utilisée pour l'extraction de structures linguistiques, à partir d'arbres d'analyse, ainsi que l'extraction de motifs dans des recettes de cuisine [Ferré et Cellier, 2016, 2022]. GCA a également été utilisée pour traiter les problèmes d'alignement des graphes de connaissances [Ferré, 2022], qui apparaissent lors de la fusion de différents graphes de connaissances pour un domaine donné.

Dans ce chapitre, nous détaillons les notions et principes sur lesquels repose GCA pour le calcul des concepts dans les graphes. Après quelques préliminaires techniques présentés en section 4.2, la section 4.3 introduit les notions de contexte graphe et de *graph pattern*, qui constituent des éléments fondamentaux de GCA. La section 4.4 poursuit avec la définition des concepts graphes, en s'appuyant sur les notions d'extension et d'intension au sens de GCA. L'algorithme de calcul des k-concepts est ensuite décrit en section 4.5. Enfin, la section 4.6 aborde la notion de concepts automorphes.

4.2 Préliminaires : tuples et projections

Dans cette section, nous présentons les notions de *tuples* et de *projections* qui sont des concepts mathématiques importants pour Graph-FCA.

Tuples. Un tuple est une séquence ordonnée d'éléments noté $\overline{x} = (x_1, \ldots, x_k)$, où $|\overline{x}| = k$ est son arité. Par exemple, un enregistrement dans une base de données peut être représenté par le tuple $\overline{x}$ = (*Alice, female, city, single*) qui identifie une personne et ses caractéristiques. L'ensemble de tous les k-tuples sur un domaine E est noté E^k et l'ensemble de tous les tuples, indépendamment de leur arité, est défini par $E^* = \bigcup_{k\geq 0} E^k$. Il n'existe qu'un unique 0-tuple, noté $()$. Nous utilisons $1..k$ pour désigner l'ensemble des entiers de 1 à k. Afin d'éviter toute confusion avec d'autres types d'indices, la notation $\overline{x}[i]$ peut être utilisée comme alternative à x_i. Il est courant d'appliquer des fonctions à des tuples. Pour toute fonction unaire ϕ, la notation $\phi(\overline{x})$ désigne le tuple $(\phi(\overline{x}[1]), \phi(\overline{x}[2]), \ldots, \phi(\overline{x}[k]))$ et on a $\phi(\overline{x})[i] = \phi(x[i])$; et pour toute fonction binaire, la notation $\psi(\overline{x}, \overline{y})$ désigne le tuple tel que $\psi(\overline{x}, \overline{y})[i] = \psi(\overline{x}[i], \overline{y}[i])$.

Projections. La projection est une opération permettant de sélectionner ou d'extraire une

partie spécifique d'un ensemble ou d'une structure. La projection sur un tuple signifie extraire certains éléments spécifiques de celui-ci. Formellement, une projection $\pi \in \Pi_k^l$ est utilisée pour faire correspondre un k-tuple à un l-tuple selon la formule suivante : $\pi(\overline{x})[i] = \overline{x}[\pi(i)]$, c'est-à-dire que le i-ième élément d'un tuple projeté est l'élément à l'indice $\pi(i)$. Elle est définie comme une fonction $\pi : \{1, \ldots, l\} \to \{1, \ldots, k\}$ reliant les indices $1..l$ du tuple cible aux indices $1..k$ du tuple source. Pour illustration, la projection $\pi = \{1 \mapsto 4, 2 \mapsto 1\}$ fait correspondre un 4-tuple (ou un tuple d'arité ≥ 4) à un 2-tuple où le premier élément du tuple cible (de sortie) est le quatrième élément du tuple source (d'entrée), et le deuxième élément du tuple cible est le premier élément du tuple source. La projection π peut être représentée de manière plus concise par le tuple $(4, 1)$. Soient le tuple source $\overline{x}$ = (*Alice, female, city, single*) et la projection $\pi = (4, 1)$. Le tuple projeté $\overline{y} = \pi(\overline{x})$ a pour composantes $\overline{y}[1] = \overline{x}[4]$ et $\overline{y}[2] = \overline{x}[1]$, donc $\overline{y} = \pi(\overline{x})$ = (*single, Alice*). Ainsi, l'élément à l'indice i du tuple projeté $\overline{y}$ est l'élément à l'indice $\pi(i)$ dans le tuple source $\overline{x}$.

4.3 Contexte graphe et *graph pattern*

Cette section fournit une définition des données d'entrée de GCA, représentées sous la forme d'un contexte graphe (*graph context*), et introduit la notion de *graph patterns*, qui permettent de capturer des motifs structurels présents dans ces données.

4.3.1 Contexte formel en Graph-FCA : contexte graphe

Dans GCA, les objets sont représentés par des nœuds du graphe, les relations par des arêtes orientées entre ces nœuds, et les attributs par des étiquettes associées aux nœuds et aux arêtes. Alors que l'AFC définit ses données d'entrée comme un contexte formel qui est une relation d'incidence entre les objets et les attributs, GCA définit ses données d'entrée comme un contexte graphe qui est une relation d'incidence entre les **tuples d'objets** et les attributs.

Définition 4.1 (Contexte graphe). Un contexte graphe est un triplet $K = (O, A, I)$, où O est un ensemble d'objets, A un ensemble d'attributs et $I \subseteq O^* \times A$ une relation d'incidence entre les tuples d'objets $\overline{o} \in O^* = \bigcup_{k \geq 1} O^k$ et les attributs $a \in A$.

Une incidence $((o), a)$ décrit l'objet o par l'attribut a comme dans l'AFC. Une incidence $((o_1, o_2), a)$ relie l'objet $o1$ à l'objet o_2 par une relation binaire a comme dans RCA. Une incidence $((o_1, \ldots, o_n), a)$ représente une relation n-aire. Pour plus de lisibilité, une alternative à la notation $((o_1, \ldots, o_n), a)$ peut être : $a(o_1, \ldots, o_n)$ comme en logique des prédicats. Les attributs unaires sont utilisés pour étiqueter les nœuds et les attributs n-aires sont utilisés pour étiqueter les arêtes reliant n nœuds. Dans un contexte graphe, chaque

nœud est représenté par un rectangle ayant deux compartiments où le premier compartiment identifie l'objet et le second compartiment représente la description de l'objet. Pour illustration, la figure 4.1 fournit un petit contexte graphe sur la famille royale britannique. Dans ce contexte graphe, l'incidence unaire *male*($Georges$) décrit l'entité $Georges$ par l'attribut *male* et l'incidence binaire *has-parent*($Georges, William$) signifie que $Georges$ *a pour parent (has-parent)* $William$.

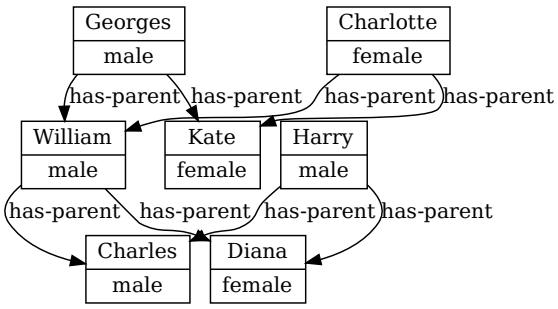

Figure 4.1 – Contexte graphe sur la famille royale britannique [Ferré et Cellier, 2016].

Afin d'identifier les représentions extensionnelles et intensionnelles de GCA, nous devons commencer par nous demander ce qu'est une description adéquate d'un objet dans GCA. Un objet (par exemple, $Georges$) doit au moins être décrit par ses arêtes adjacentes c'est-à-dire, les arêtes $a(\overline{o})$, où $\overline{o}$ est un tuple d'objets contenant au moins l'objet concerné (par exemple, *has-parent*($Georges, William$)). Ensuite, si les objets adjacents sont liés à d'autres objets (par exemple, *has-parent*($William, Charles$)), cela doit également apparaître dans la description de l'objet. En effet, les descriptions des objets adjacents doivent être incluses, car elles ont un impact indirect sur la nature de l'objet à décrire. En tout, cela implique que la description d'un objet est l'ensemble du graphe, ou au moins la composante connexe à laquelle il appartient si le graphe est composé d'un ensemble de sous-graphes. Ainsi, dans un contexte graphe $K = (O, A, I)$, la description d'un objet o peut être définie comme le couple (o, I). De la même manière, la description d'un tuple d'objets $\overline{o}$ peut être définie par le couple $(\overline{o}, I)$ que nous notons $Q(\overline{o})$.

Nous présentons dans la section suivante les *graph patterns* qui constituent une généralisation de ces descriptions d'objets et permettent en GCA de définir des représentations intensionnelles sur des ensembles de tuples d'objets, plutôt que sur des ensembles d'objets.

4.3.2 Graph patterns

Les descriptions d'objets doivent être généralisées pour former les descriptions partagées par un ensemble d'objets ou par un ensemble de tuples d'objets, d'où la notion de

graph patterns comme généralisation des contextes graphes. GCA définit un *graph pattern* comme une relation d'incidence généralisée, et un *Projected Graph Pattern* (PGP) comme une description généralisée. Dans les définitions suivantes, $K = (O, A, I)$ est un contexte graphe et $\mathcal{V}$ est un ensemble de variables tel que $O \subseteq \mathcal{V}$.

Définition 4.2 (Graph pattern). Un graph pattern $P \subseteq \mathcal{V}^* \times A$ est un ensemble d'arêtes n-aires avec des variables comme nœuds et des attributs comme étiquettes. En termes simples, un graph pattern constitue une abstraction d'un contexte graphe, où les objets sont généralisés sous forme de variables. $V(P)$ désigne l'ensemble des variables de P.

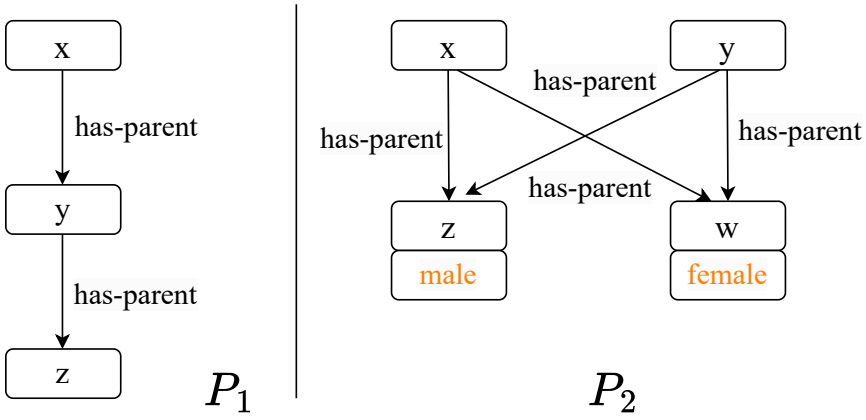

FIGURE 4.2 – Représentation graphique des graph patterns P_1 et P_2.

Pour illustration, le graph pattern $P_1 = \{\textit{has-parent}(x, y), \textit{has-parent}(y, z)\}$ décrit toute situation où "une entité x a pour parent (*has-parent*) une autre entité y, qui a un parent z". Le pattern $P_2 = \{\textit{has-parent}(x, z), \textit{has-parent}(x, w), \textit{has-parent}(y, z), \textit{has-parent}(y, w), \textit{male}(z), \textit{female}(w)\}$ est un graph pattern qui décrit une situation où "deux entités x et y ont le même père et la même mère", c'est-à-dire que x et y sont frères/soeurs. La figure 4.2 présente les patterns P_1 et P_2 sous forme graphique. Notons que le pattern P_1 peut être vu comme une généralisation du contexte graphe présenté à la figure 4.3 qui est extrait du contexte de la figure 4.1. Dans P_1, le genre (*male*/*female*) des personnes n'est pas pris en compte, ce qui explique pourquoi la variable x de P_1 matche avec $Georges$ et $Charlotte$. Il en est de même pour la variable z qui a pour instances $Diana$ et $Charles$.

4.3.2.1 *Pattern core* (motif principal) d'un graph pattern

En termes de redondances d'information dans un graphe, une notion importante en théorie des graphes est celle du *core* (noyau) d'un graphe, qui correspond à la version mi-

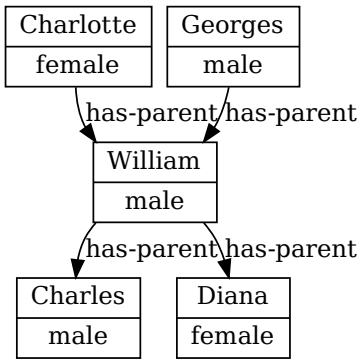

Figure 4.3 – Un sous-graphe du contexte graphe de la figure 4.1.

nimale de ce graphe qui exprime la même information que le graphe initial. Il est en effet naturel de se demander lorsque l'on examine des sous-structures présentant une certaine propriété, s'il existe une sous-structure minimale possédant cette même propriété. Cette notion de version minimale d'un graphe (graph pattern dans notre cas) est utile pour une description (intension) non redondante d'un ensemble de tuples d'objets.

Le *core* d'un graphe est son plus petit *retract* et le *retract* d'un graphe G est un sous-graphe H de G au sens de l'homomorphisme c'est-à-dire, qui respecte la structure de graphe [Hahn et Tardif, 1997].

Définition 4.3 (Retract d'un graphe). Soient G et H deux graphes. H est appelé un retract de G s'il existe deux homomorphismes $\rho : G \to H$ et $\gamma : H \to G$ tels que $\rho \, o \, \gamma = id_H$. L'homomorphisme ρ est appelé une rétraction et γ une co-rétraction [Hahn et Tardif, 1997].

Définition 4.4 (Core d'un graphe). Un graphe G est un core si aucun sous-graphe propre [1] de G n'est un retract de G [Hahn et Tardif, 1997].

En d'autres termes, le *core* d'un graphe est sa forme la plus compacte du point de vue de ses relations de voisinage c'est-à-dire, sans informations redondantes et qui capture la structure essentielle du graphe. Ainsi, on dit d'un graphe qu'il est son propre *core* s'il n'a aucune simplification possible par homomorphisme. Pour illustrer, la figure 4.4 reprend l'exemple utilisé dans [Ferré et Cellier, 2020].

Dans la figure 4.4, les graphes **H** et **G1** sont des retracts du graphe **G2** mais pas du graphe **G3**. En effet, affirmer plusieurs fois que x est dans une relation a avec quelque

1. Un sous-graphe propre d'un graphe G est un sous-graphe qui n'est pas identique à G lui-même.

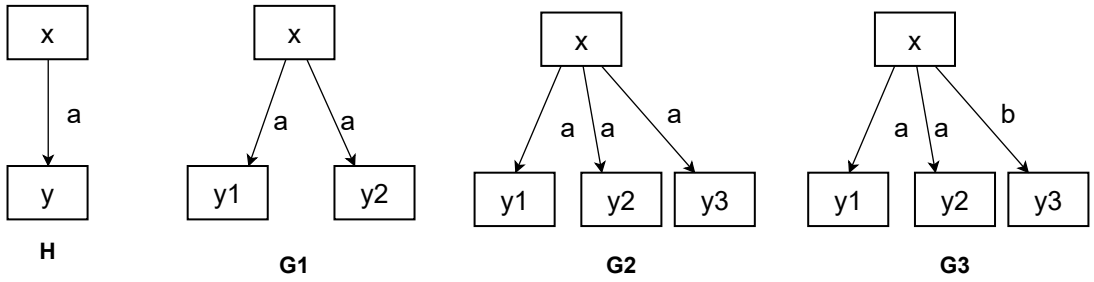

FIGURE 4.4 – Les graphes **H** et **G1** sont des retracts du graphe **G2** mais pas du graphe **G3**. Le graphe **H** est le core des graphes **G1** et **G2**. Par conséquent, les graphes **H**, **G1** et **G2** sont équivalents.

chose n'ajoute rien à l'affirmer une seule fois. Effectivement, les variables $y1$, $y2$ et $y3$ de **G2** peuvent correspondre au même objet dans le contexte graphe, car aucune information (par exemple, une étiquette) ne les différencie. Au contraire, **G3** affirme que x est à la fois dans une relation a et une relation b, et ne peut donc pas se rétracter en H : l'arête $b(x, y3)$ ne peut pas se replier sur l'arête $a(x, y2)$. Ainsi, le graphe **H** est le core des graphes **G1** et **G2** mais pas de **G3**.

4.3.2.2 Projected Graph Pattern

Un graph pattern représente un motif de données présent dans le contexte graphe. L'opération permettant de trouver les instances d'un tel motif consiste à projeter les variables du pattern dans le contexte graphe pour trouver ses différentes instances. Autrement dit, il s'agit d'identifier, dans le contexte graphe, les objets susceptibles de substituer les variables du pattern. Le résultat obtenu est appelé un *Projected Graph Pattern* (PGP).

Définition 4.5 (Projected Graph Pattern-PGP). Un PGP est un couple $Q = (\overline{x}, P)$ où P est un graph pattern, et $\overline{x} \in \mathcal{V}^*$ appelé tuple de projection, est un tuple de variables. Autrement dit, un PGP est un focus sur un ou plusieurs nœuds d'un graph pattern. Les projections sur les tuples sont étendues aux PGPs : $\pi(Q) = (\pi(\overline{x}), P)$. $|Q| = |\overline{x}|$ désigne l'arité du PGP et on parle de k-PGP pour représenter un PGP d'arité k.

Un tuple de projection peut être vu comme un tuple d'objets abstraits par des variables ; il définit un focus sur les nœuds du graph pattern. À titre d'exemple, dans le figure 4.2, le PGP $Q_1 = ((x), P_1)$ qui a pour tuple de projection (x) - i.e, avec un focus sur la variable x de P_1 - décrit toute entité x ayant un grand-parent z, inversement, le PGP $Q_2 = ((z), P_1)$ décrit toute entité z ayant un petit-enfant x. Un autre exemple est le PGP $Q = ((x, y), P_2)$ qui décrit quant à lui, la relation de "frère/soeur" entre x et y. On peut voir le PGP comme

une requête que l'on évalue sur le contexte graphe. À titre d'illustration, voici quelques projections des patterns P_1 et P_2 par rapport au contexte graphe de la figure 4.1. Ces exemples de projections sont représentés schématiquement dans la figure 4.5.

- $Q_1 = ((x), P_1)$ a pour résultat les objets suivants : $Georges$ et $Charlotte$ (qui ont pour grands-parents $Charles$ et $Diana$).
- $Q_2 = ((z), P_1)$ a pour résultat les objets (grands-parents) $Charles$ et $Diana$.
- $Q = ((x, y), P_2)$ a pour résultat $(Georges, Charlotte)$ et $(William, Harry)$ qui sont des couples[2] représentant des frères/soeurs ayant le même père et la même mère. En effet, il y a d'une part $Georges$ et $Charlotte$ qui ont pour parents $William$ et $Kate$, et d'autre part $William$ et $Harry$ qui ont pour parents $Charles$ et $Diana$.

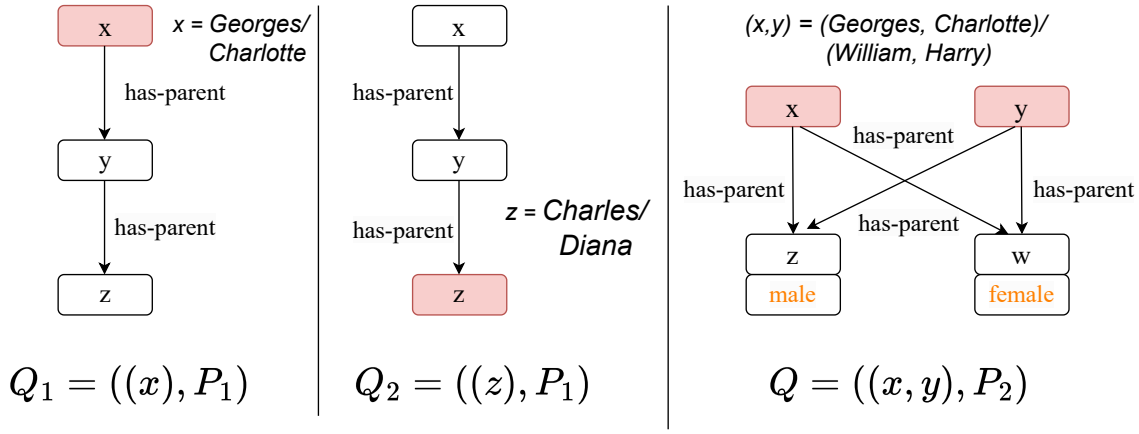

Figure 4.5 – Représentation graphique des PGPs Q_1, Q_2 et Q par rapport au contexte graphe de la figure 4.1.

Les arêtes des patterns peuvent être vues comme des contraintes sur les variables. Une variable qui apparaît dans le tuple de projection mais pas dans le pattern n'est pas contrainte et peut prendre n'importe quel objet comme valeur. Une variable qui apparaît dans le pattern mais pas dans le tuple de projection est quantifiée de manière existentielle par rapport aux variables projetées. Par exemple dans $Q_1 = ((x), P_1)$, il doit exister une personne (z) qui soit grand-parent de x. Dans $Q = ((x, y), P_2)$, il doit exister une entité z qui soit le père de x et y et une entité w qui soit leur mère ; mais les identités de z et w n'ont pas d'importance. Plus précisément, un PGP décrit son tuple de projection, et par conséquent, l'ensemble de ses instances.

Dans la section suivante, nous présentons la notion de concept graphe (*graph concept*) équivalente à la notion de concept formel en AFC à travers les définitions d'une extension

2. En effet, le tuple de projection contient 2 éléments. La nature du tuple de projection détermine la nature des résultats de la projection.

et d'une intension de concept en GCA.

4.4 Extension, intension et concept graphe

Cette section introduit les représentations intensionnelles et extensionnelles qui permettent de définir un concept graphe GCA.

4.4.1 Extension : des PGPs aux *object relations*

Dans la description des PGPs donnée précédemment, nous avons vu que le résultat d'une projection est un ensemble de tuples d'objets (de même arité que le tuple de projection) contraint par les caractéristiques définies par le PGP. Ceci permet de définir les représentations extensionnelles de GCA comme ensembles de tuples d'objets avec la contrainte que tous les tuples d'un même ensemble aient la même arité. Ainsi, nous obtenons que les extensions sont des *object relations*, c'est-à-dire des ensembles de tuples d'objets.

Définition 4.6 (Object relation). Un *object relation* est un ensemble de tuples d'objets $R \subseteq O^k$, pour une certaine arité $|R| = k$. On note $\mathcal{R}$ l'ensemble des object relations indépendamment de leur arité et $\mathcal{R}_k$ le sous-ensemble des relations d'arité k. Les projections sont étendues aux relations : $\pi(R) = \{\pi(\overline{o}) \mid \overline{o} \in R\}$.

Par exemple, $\{(William, Harry), (Georges, Charlotte), (Diana, Kate)\}$ est un object relation d'arité 2. Un object relation d'arité 1 est tout simplement équivalent à un ensemble d'objets comme en AFC.

L'obtention des extensions de concepts en GCA se fait par une correspondance des PGPs aux ensembles de tuples d'objets qui détermine les instances des tuples de projection sur un contexte graphe donné. Notamment, le fait que le focus sur le nœud x du pattern P_1 ($Q_1 = ((x), P_1)$) fournit $Georges$ et $Charlotte$ comme instances de x indique qu'il y a au moins 2 instances de P_1 (du point de vue de x) dans le contexte graphe.

L'opération d'inclusion sur les ensembles d'attributs en AFC s'étend aux PGPs en GCA pour la définition des extensions.

Définition 4.7 (Inclusion de PGP). Soient $Q_1 = (\overline{x}_1, P_1)$ et $Q_2 = (\overline{x}_2, P_2)$ deux PGPs de même arité ($|Q_1| = |Q_2|$). Q_1 est inclus dans Q_2 et on note $Q_1 \subseteq_q Q_2$ si et seulement si, il existe un homomorphisme[3] $\phi : V(P_1) \to V(P_2)$ tel que : $\phi(\overline{x}_1) = \overline{x}_2$ et $\phi(P_1) \subseteq P_2$ i.e., $\forall(\overline{y}, a) \in P_1, (\phi(\overline{y}), a) \in P_2$.

3. ϕ est donc un morphisme de graphes et on dit classiquement que P_1 se "projette" dans P_2 avec conservation des tuples de projections.

Définition 4.8 (Extension d'un PGP). Soit $K = (O, A, I)$ un contexte graphe. L'extension d'un k-PGP $Q = (\overline{x}, P)$, notée $ext(Q)$, est définie par

$$ext(Q) := \{\overline{o} \in O^k \mid Q \subseteq_q Q(\overline{o})\}, \text{ où } Q(\overline{o}) = (\overline{o}, I)$$

Littéralement, l'extension d'un k-PGP Q est l'ensemble de k-tuples d'objets contenant Q (qui partagent Q) dans leur description respective, sachant que chaque tuple d'objets $\overline{o}$ a pour description $Q(\overline{o}) = (\overline{o}, I)$. Donc, pour chaque tuple d'objets $\overline{o} \in O^k$, Q est une généralisation de sa description $Q(\overline{o})$ c'est-à-dire, une version obtenue en substituant certains objets par des variables et en relâchant certaines contraintes.

Pour clarifier, par rapport au contexte graphe de la figure 4.1, le PGP $Q = ((x, y), P_2)$ avec $P_2 = \{$*has-parent*(x, z), *has-parent*(x, w), *has-parent*(y, z), *has-parent*(y, w), *male*(z), *female*$(w)\}$, a pour extension $ext(Q) = \{(William, Harry), (Georges, Charlotte)\}$. De même, le PGP $Q' = ((x, y), P_2')$, où $P_2' = \{$*has-parent*(x, z), *has-parent*$(x, Diana)$, *male*(z), *has-parent*(y, z), *has-parent*$(y, Diana)$, *female*$(Diana)\}$, possède pour extension $ext(Q') = \{(William, Harry)\}$, qui est plus spécifique que celle de Q, c'est-à-dire $ext(Q') \subseteq ext(Q)$. Ce résultat est normal car le PGP Q décrit l'ensemble des paires de personnes ayant le même père et la même mère, sans préciser leur identité, tandis que Q' contraint la description en spécifiant que la mère est $Diana$.

Formellement, $ext(Q) = ext(((x, y), P_2)) = \{\overline{o} \in O^2 \mid Q \subseteq_q (\overline{o}, I)\}$. Pour illustrer cette définition, nous montrons un couple $\overline{o} \in O^2 \mid Q \subseteq_q (\overline{o}, I)$ et un autre couple tel que $Q \nsubseteq_q (\overline{o}, I)$. Pour $\overline{o} = (Georges, William)$, on a $Q \nsubseteq_q ((Georges, William), I)$ comme illustré à la figure 4.6 car, il n'y a pas de matching/correspondance entre les 2 tuples de projections (x, y) et $(Georges, William)$. En effet, $Georges$ et $William$ n'ont pas les mêmes parents comme c'est le cas pour x et y.

Pour $\overline{o} = (William, Harrry)$, on obtient que $((x, y), P_2) \subseteq_q ((William, Harrry), I)$ comme le montre la figure 4.7. En effet, il existe un homomorphisme ϕ qui projette chaque nœud et arête de P_2 dans I, avec correspondance entre (x, y) et $(William, Harrry)$ qui sont les tuples de projections. Par conséquent, $(William, Harrry)$ est un élément de l'extension de Q, car est une instance de son tuple de projection (x, y). Ce qui est également le cas pour $(Georges, Charlotte)$.

4.4.2 Intension : des *objects relations* aux PGPs

Tout comme l'AFC définit une correspondance des sous-ensembles d'objets vers les sous-ensembles d'attributs, GCA définit une correspondance des *objects relations* (ensemble de tuple d'objets) aux PGPs (intensions). Pour chaque object relation, il définit son intension comme l'intersection des descriptions (PGPs) de tous les tuples d'objets qu'il contient.

L'intersection de PGPs $\cap_q$ est définie comme une forme d'alignement de graphes, où

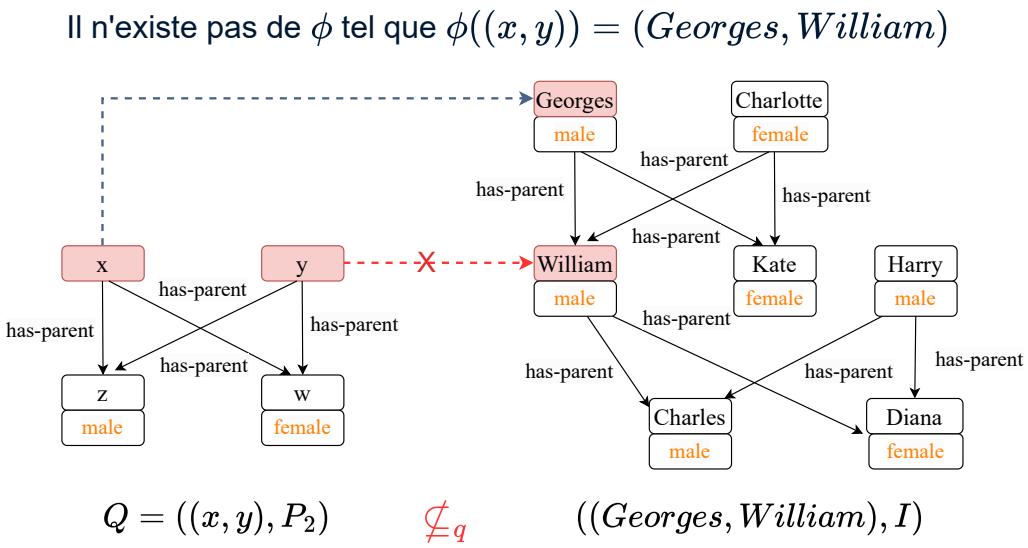

FIGURE 4.6 – $Q = ((x,y), P_2) \not\subseteq_q ((Georges, William), I)$.

chaque paire de variables des deux patterns devient une variable du pattern d'intersection. Elle correspond au produit catégorique des graphes ([HAHN et TARDIF, 1997]), également appelé produit tensoriel ou produit direct. Le produit catégorique $G \times H$ de deux graphes G et H est donné par :

$$E(G \times H) = \{[(u,x),(v,y)] : [u,v] \in E(G), [x,y] \in E(H)\}.$$

Définition 4.9 (Intersection de PGPs). Soit ψ une injection de $\mathcal{V}\times\mathcal{V}$ dans $\mathcal{V}$. L'intersection de deux k-PGPs $Q_1 = (\overline{x}_1, P_1)$ et $Q_2 = (\overline{x}_2, P_2)$, noté $Q_1 \cap_q Q_2$ est définie par $Q = (\overline{x}, P)$, où

- $\overline{x} = \psi(\overline{x}_1, \overline{x}_2)$,
- $P = P_1 \times P_2 = \{(\psi(\overline{y}_1, \overline{y}_2), a) \,|\, a \in A, (\overline{y}_1, a) \in P_1,\ (\overline{y}_2, a) \in P_2,\ |\overline{y}_1| = |\overline{y}_2|\}$.

À titre d'illustration, soient $Q_1 = ((x_1), P_1)$ avec $P_1 = \{a(x_1, y_1), c(x_1, z_1), b(y_1, z_1)\}$ et $Q_2 = ((x_2), P_2)$ avec $P_2 = \{a(x_2, y_2), c(x_2, y_2), b(y_2, y_2)\}$ deux PGPs, calculons leur intersection $Q = Q_1 \cap_q Q_2 = ((x), P)$. La figure 4.8 explicite les représentations graphiques de Q_1 et Q_2. Le calcul de Q s'obtient par produit catégorique $P = P_1 \times P_2$ et le tuple de projection s'obtient par conservation et par substitution de variables. La figure 4.9 présente les nœuds et les arêtes (relations) de Q. Les cases vides du tableau matérialisent l'absence d'arêtes entre les sommets concernés, et les cases non vides contiennent l'attribut de l'arête entre les deux sommets correspondants. Sur la base de ces informations, la figure 4.10 illustre la représentation graphique de l'intersection $Q = Q_1 \cap_q Q_2$ ayant pour expression algébrique : $Q = ((x), P) = ((x), \{a(x,y), c(x,z), b(y,z)\})$.

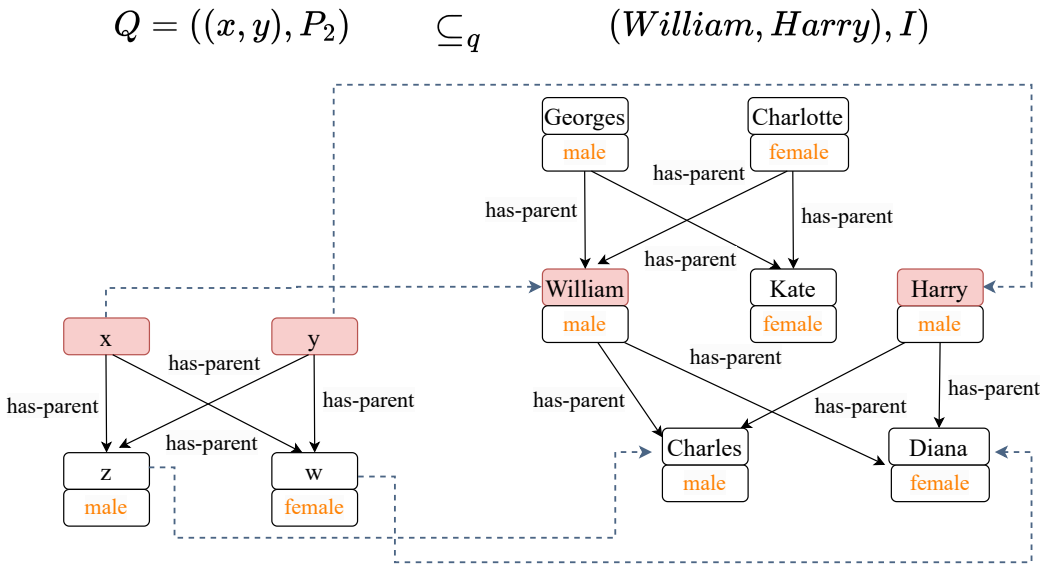

Figure 4.7 – $Q = ((x, y), P_2) \subseteq_q ((William, Harry), I)$.

Cette notion d'intersection de PGPs (intersection des descriptions) permet de calculer les intensions de concepts en GCA, tout comme le calcul d'intensions de concepts en AFC qui se fait par intersections d'ensembles d'attributs.

Définition 4.10 (Intension). Soit $K = (O, A, I)$ un contexte graphe. L'intension d'un *object relation* $R \in \mathcal{R}_k$, noté $int(R)$ est définie par

$$int(R) = \bigcap_{\overline{o} \in R} Q(\overline{o})$$

4.4.3 Concept graphe

En Graph-FCA, les intensions de concepts sont des Projected Graph Patterns (PGP) et les extensions sont les ensembles de tuples d'objets (*object relations*). Une intension de concept est un PGP qui décrit tout ce qu'un ensemble de tuples d'objets a en commun. Dans un cas particulier, l'intension d'un objet unique est la description de cet objet.

Sur la base des définitions précédentes, la connexion de Galois suivante peut être définie. Les correspondances définies des PGPs aux *objects relations* et inversement des *objects relations* vers les PGPs forment la connexion de Galois qui fonde la définition d'un concept graphe (preuve dans [Ferré et Cellier, 2020]).

Théorème 1 (Connexion de Galois). Soit $K = (O, A, I)$ un contexte graphe. Pour chaque arité k, la paire de correspondances (ext, int) forme une connexion de Galois entre $(\mathcal{R}_k, \subseteq)$

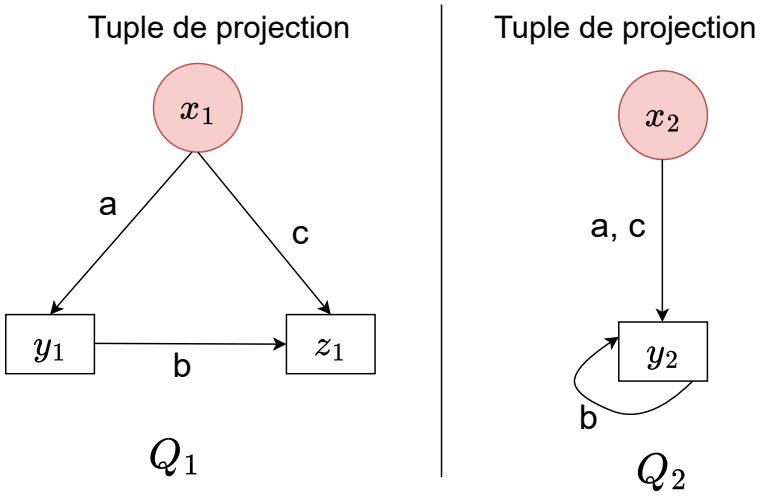

FIGURE 4.8 – Représentation graphique des PGPs Q_1 (à gauche) et Q_2 (à droite).

et $(\mathcal{Q}_k, \subseteq_q)$ avec $ext(Q) := \{\overline{o} \in O^k \,|\, Q \subseteq_q (\overline{o}, I)\}$ et $int(R) := \sqcap_q\{(\overline{o}, I)\}_{\overline{o}\in R}$. En d'autres termes, pour chaque object relation $R \in \mathcal{R}_k$ et pour chaque PGP $Q \in \mathcal{Q}_k$,

$$R \subseteq ext(Q) \iff Q \subseteq_q int(R)$$

L'expression $ext(Q)$ traduit le fait que l'extension d'une description (attribut au sens de l'AFC) soit égale à l'ensemble des tuples d'objets contenant cette description (Q) dans leur propre description. L'expression duale $int(R)$ quant à elle, représente le fait que la description d'un ensemble de tuples d'objets soit égale, à l'intersection des descriptions de chaque tuple d'objets ; ceci pour trouver la description qui est commune (comme l'intersection des ensembles d'attributs en AFC). À partir de la connexion de Galois (théorème 1), les concepts graphes peuvent être calculés et organisés en treillis de concepts comme en AFC classique pour chaque arité k.

Définition 4.11 (Concept graphe). Soit $K = (O, A, I)$ un contexte graphe. Un concept graphe de K est une paire (R, Q) constituée d'un ensemble de tuples d'objets (extension) et d'un PGP (intension) tels que $R = ext(Q)$ et $Q \equiv_q int(R)$. L'arité d'un concept graphe est l'arité de son extension et de son intension, qui doivent être égales.

4.5 Calcul et représentation des k-concepts

Maintenant que la notion de concept graphe a été introduite, détaillons le processus algorithmique permettant de calculer l'ensemble des k-concepts pour un contexte graphe donné (section 4.5.1), ainsi que leur représentation graphique (section 4.5.2).

Arêtes de Q_1 :

$\{a(x_1, y_1), c(x_1, z_1), b(y_1, z_1)\}$

Arêtes de Q_2 :

$\{a(x_2, y_2), c(x_2, y_2), b(y_2, y_2)\}$

Noeuds de $Q = Q_1 \cap_q Q_2 = noeuds - Q_1 \times noeuds - Q_2$

$= \{(x_1, x_2), (x_1, y_2), (y_1, x_2), (y_1, y_2), (z_1, x_2), (z_1, y_2)\}$

Calcul des arêtes de Q

	(x_1, x_2)	(x_1, y_2)	(y_1, x_2)	(y_1, y_2)	(z_1, x_2)	(z_1, y_2)
(x_1, x_2)				$\{a\} \cap \{a, c\}$		$\{c\} \cap \{a, c\}$
(x_1, y_2)						
(y_1, x_2)						
(y_1, y_2)						$\{b\} \cap \{b\}$
(z_1, x_2)						
(z_1, y_2)						

Figure 4.9 – Calcul des nœuds et arêtes de $Q = Q_1 \cap_q Q_2$.

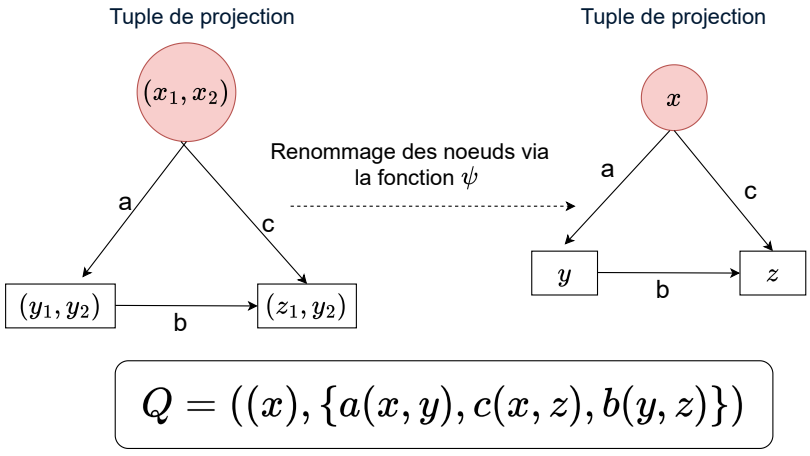

Figure 4.10 – Représentation algébrique et graphique de $Q = Q_1 \cap_q Q_2$.

4.5.1 Algorithme de calcul des k-concepts

L'algorithme 4 synthétise l'algorithme de calcul des k-concepts décrit en détail dans [Ferré et Cellier, 2020]. Il est important de noter que l'appellation k-concept signifie, concept d'arité k, c'est-à-dire un concept dont l'extension est un ensemble de tuples d'objets d'arité k. Pour $k = 1$, on parle de concept unaire où une extension de concept est un

Algorithm 4 Génération des k-concepts pour $k \in 1..Z$

Input : Un contexte graphe $K = (O, A, I)$

1: $PatternBasis \leftarrow \varnothing$ // base des motifs (patterns) regroupés par *core*
2: $Concepts \leftarrow \varnothing$; $Patterns \leftarrow \{I\}$
3: **for** $P \in Patterns$ **do**
4: $\quad Patterns \leftarrow Patterns \setminus \{P\}$
5: $\quad NewPatterns \leftarrow GenerationOfGraphPatterns(P \times I)$
6: $\quad$ **for** $P_a \in NewPatterns$ (non encore traités modulo isomorphisme) **do**
7: $\quad\quad \mathrm{P}_c \leftarrow$ le *core* de P_a
8: $\quad\quad PattenBasis \leftarrow PatternBasis \cup \{P_a @ \mathrm{P}_c\}$
9: $\quad\quad$ **for** $k \in 1...Z, \overline{x} \in V(P_a)^k$ **do**
10: $\quad\quad\quad Concepts \leftarrow Concepts \cup ConceptComputation(k, \overline{x})$
11: $\quad\quad$ **end for**
12: $\quad\quad Patterns \leftarrow Patterns \cup \{P_a\}$
13: $\quad$ **end for**
14: **end for**

ensemble d'objets[4] (comme en ACF et en RCA). Pour $k = 2$, on parle de concepts binaires (ou concept de relations binaires) où une extension de concept est un ensemble de couples d'objets. Par généralisation, on parle de concepts n-aires lorsque $k = n$. Par souci de simplicité, les illustrations de cet algorithme se limiteront au cadre des concepts unaires. L'algorithme 4 prend en entrée un contexte graphe $K = (O, A, I)$ et calcule l'ensemble des k-concepts suivant les étapes ci-dessous.

Génération de graph patterns (ligne 5). Le premier ensemble de patterns est calculé en effectuant le produit catégorique $I \times I$. Chaque composante connexe (P_a) issue de ce produit est considérée comme un pattern indépendant. Une étape d'optimisation est ensuite mise en œuvre pour simplifier les patterns P_a en cas de nœuds dupliqués. Deux nœuds d'un pattern sont dits dupliqués (ou équivalents) lorsqu'ils jouent exactement le même rôle dans ce pattern, c'est-à-dire lorsqu'ils possèdent les mêmes liaisons/connexions. Par exemple, dans le graphe **G1** à la figure 4.4, les nœuds $y1$ et $y2$ illustrent des nœuds dupliqués.

Grouper les patterns par *core* (lignes 7 et 8). Cette étape consiste, pour chaque pattern P_a, à calculer son *pattern core* P_c (**ligne 7**), puis à le rattacher au pattern partageant le même *core*, s'il existe déjà ; sinon c'est un nouveau pattern qui est rajouté à la base de patterns (**ligne 8**). Ce regroupement permet d'obtenir une représentation visuelle compacte.

Calcul des k-concepts pour chaque pattern P_a (ligne 10). Pour chaque valeur donnée de k, et pour chaque tuple de projection $\overline{x}$ du pattern P_a, le concept graphe correspondant

4. Ensemble de tuples d'objets de taille 1 plus exactement.

est obtenu par des opérations de projection et de fermeture : (1) calcul du plus petit *retract* P_r de P_a contenant le tuple de projection $\overline{x}$; (2) calcul de l'extension du concept par fermeture sur son intension, laquelle correspond au PGP $Q = (\overline{x}, P_r)$.

Itérer avec de nouveaux patterns. Les nouveaux patterns P_a obtenus comme expliqué ci-dessus deviennent une entrée pour l'étape suivante (**ligne 12**). L'algorithme s'arrête par saturation lorsque plus aucun nouveau pattern n'est généré (au sens de l'isomophisme).

De manière simple, on peut dire que l'algorithme de calcul des k-concepts se résume à calculer les ensembles de graph patterns sur les données (contexte graphe) par produits catégoriques de graphes, et de calculer ensuite les instances de ces patterns par projection de chaque nœud/variable (du graph pattern) sur le contexte graphe.

4.5.2 Représentation des résultats en Graph-FCA

GCA présente ses résultats sous différents modes. Cette section présente les résultats d'exécution de GCA sur le contexte graphe de la figure 4.1. Toutes les informations pratiques sur le fonctionnement de l'outil *gfca* qui implémente GCA sont présentées en détail dans le manuel [Ferré, 2019].

4.5.2.1 Représentation compacte : graph pattern

Le premier mode de présentation des résultats en GCA est la représentation compacte sous forme de graph patterns qui met en évidence les structures relationnelles dans les données en capturant les différentes connexions entre les concepts.

La figure 4.11 illustre l'ensemble des graph patterns (Q1 - Q4) produit par GCA sur le contexte graphe de la figure 4.1. Comme on peut le constater, les instances des nœuds ou variables n'ont pas encore été calculées. Par exemple, le pattern Q1 (en bleu) constitué d'un seul nœud ($Q1a$), représente les personnes possédant la caractéristique *male*. La question « quelles sont les personnes ayant la caractéristique *male*? » correspond à l'intension du concept. Son évaluation sur le contexte graphe permet d'identifier toutes les instances qui satisfont cette condition. L'ensemble de ces instances forme l'extension du concept.

Techniquement, dans un graph pattern, chaque nœud identifie un concept unaire (premier compartiment), comme illustré à la figure 4.12. Le deuxième compartiment contient les descriptions des objets, tandis que le troisième compartiment présente l'extension du concept, obtenue en projetant chaque nœud sur le contexte graphe. Notons que dans l'identifiant Qix d'un concept, i indique le numéro du pattern et x est la chaîne de caractères (généralement une seule lettre) qui identifie le concept par rapport au pattern. Par exemple, $Q1a$ est le premier concept du premier pattern (constitué d'un seul concept), et $Q3b$ est

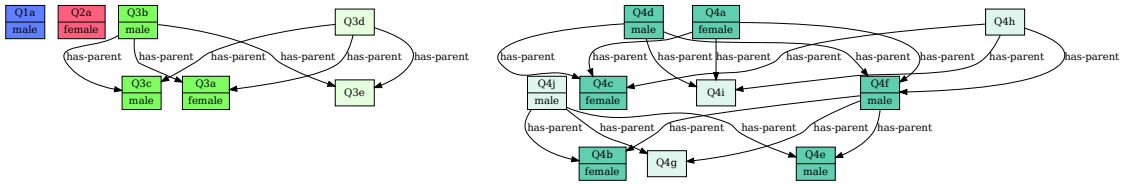

FIGURE 4.11 – Graph patterns (hors projections) du contexte graphe de la figure 4.1.

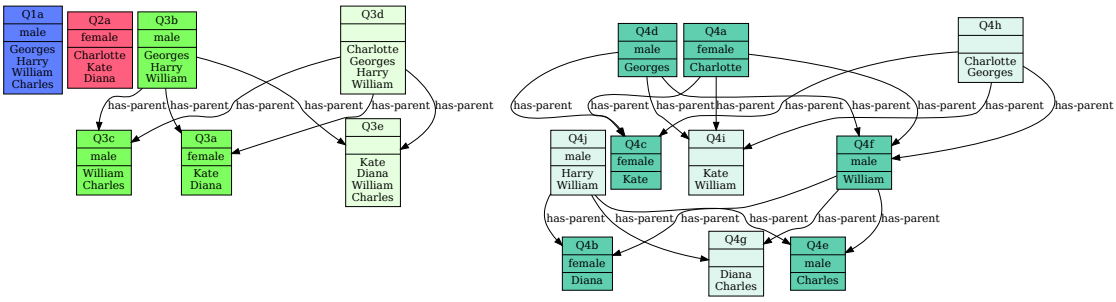

FIGURE 4.12 – Graph patterns (complet) du contexte graphe de la figure 4.1.

le deuxième concept du troisième pattern. En termes d'interprétation, $Q1a$ correspond au concept des hommes, c'est-à-dire aux individus ayant pour description *male*, tandis que $Q3b$ identifie le concept des fils, c'est-à-dire les individus décrits par l'attribut *male* et possédant au moins un parent. L'intension d'un concept Qix dans un pattern P, est donnée par le PGP $((x), Pi_r)$ où Pi_r est le sous-pattern constitué du nœud x et du pattern core de P : il s'agit du plus petit retract de P contenant le nœud x.

Dans la représentation visuelle, chaque graph pattern est illustré avec deux nuances de couleur (une couleur par pattern) : la nuance vive pour les nœuds du pattern core et la nuance claire pour les nœuds non-core. Les nœuds en nuance vive (*core nodes*) représentent les nœuds spécifiques, tandis que les nœuds en nuance claire généralisent les informations des nœuds spécifiques. En d'autres termes, le pattern core représente la sous-partie du pattern utile à la description de chaque nœud du pattern, car contenant l'essentiel de l'information contenue dans le pattern. À titre d'illustration, dans le pattern Q3 (en vert citron), le pattern core correspond au sous-graphe P_{abc}, constitué des nœuds $Q3a$, $Q3b$, et $Q3c$. Par exemple le concept de "*parent*" ($Q3d$) généralise les concepts de "*père*" ($Q3c$) et de "*mère*" ($Q3a$). Ainsi, l'intension de $Q3b$ est le PGP $((b), \mathrm{P}_{abc})$, car $Q3b$ est inclus dans le pattern core, tandis que l'intension du concept $Q3d$ est le PGP $((d), \mathrm{P}_d)$ où P_d est plus petit retract de Q_3 qui contient $Q3d$, c'est-à-dire constitué du pattern core P_{abc} et du concept $Q3d$.

4.5.2.2 Représentation hiérarchique

Un ordre de généralisation sur l'ensemble des concepts des graph patterns permet d'obtenir une représentation hiérarchique des concepts qui est semblable à l'ensemble des treillis interconnectés de RCA. Cette hiérarchie de concepts n'est rien d'autre que la représentation de la relation de subsomption[5] sur l'ensemble de concepts. Dans cette représentation, les relations entre concepts sont remplacées par des labels textuels dans la partie intensionnelle des concepts (deuxième compartiment). On obtient ainsi les concepts ordonnés des plus spécifiques aux plus généraux du bas vers le haut, ce qui permet d'avoir une vue d'ensemble de l'ordre de généralisation entre les concepts. La figure 4.13 illustre la hiérarchie des concepts des patterns de la figure 4.12. La relation de subsomption est présentée par les flèches en pointillés pointant des sur-concepts vers les sous-concepts.

Pour exemple, la relation *has-parent* entre les concepts $Q3b$ et $Q3c$ du pattern Q3 (figure 4.12) est représentée par l'attribut [*has-parent* _ c] dans l'intension du concept $Q3b$ de la hiérarchie de concepts (figure 4.13), indiquant que les entités de $Q3b$ ont leurs parents dans $Q3c$. Inversement, cette relation est représentée par l'attribut [*has-parent* b _] dans l'intension de $Q3c$, signifiant que ses entités ont leurs enfants dans $Q3b$. Ceci est rendu possible par le fait que GCA gère automatiquement les relations dans leurs deux sens (direct et indirect) ; il s'agit d'une propriété intrinsèque au produit catégorique de graphes. De manière générale, comme dans cet exemple, l'outil de GCA n'affiche pas les bottom concepts ($\bot$) qui ont une extension vide ni les top concepts ($\top$) ayant une intension vide. Notons aussi que cette représentation hiérarchique affiche l'extension et l'intension complète des concepts.

4.5.2.3 Représentation combinée : graph patterns + hiérarchie de concepts

Le troisième mode de représentation des concepts graphes est une représentation qui combine les représentations hiérarchique et graphique (graph patterns). Cette représentation combinée est la plus riche car met en évidence les connexions entre les concepts (vue pattern) et l'ordre de généralisation sur l'ensemble des concepts. Il convient de noter que cette représentation combinée est plus difficile à appréhender, bien qu'elle reste plus riche et plus complète.

La figure 4.14 fournit la représentation combinée des concepts graphes calculés sur le contexte graphe de la figure 4.1. Comme son nom l'indique, cette représentation combine celle de la figure 4.12, où les relations entre les concepts sont indiquées par des arrêtes orientées et étiquetées (en solide), et celle de la figure 4.13 où la relation de subsomption est représentée par des flèches en pointillés.

5. Un concept C est subsumé par un concept D lorsque $ext(C) \subseteq ext(D)$.

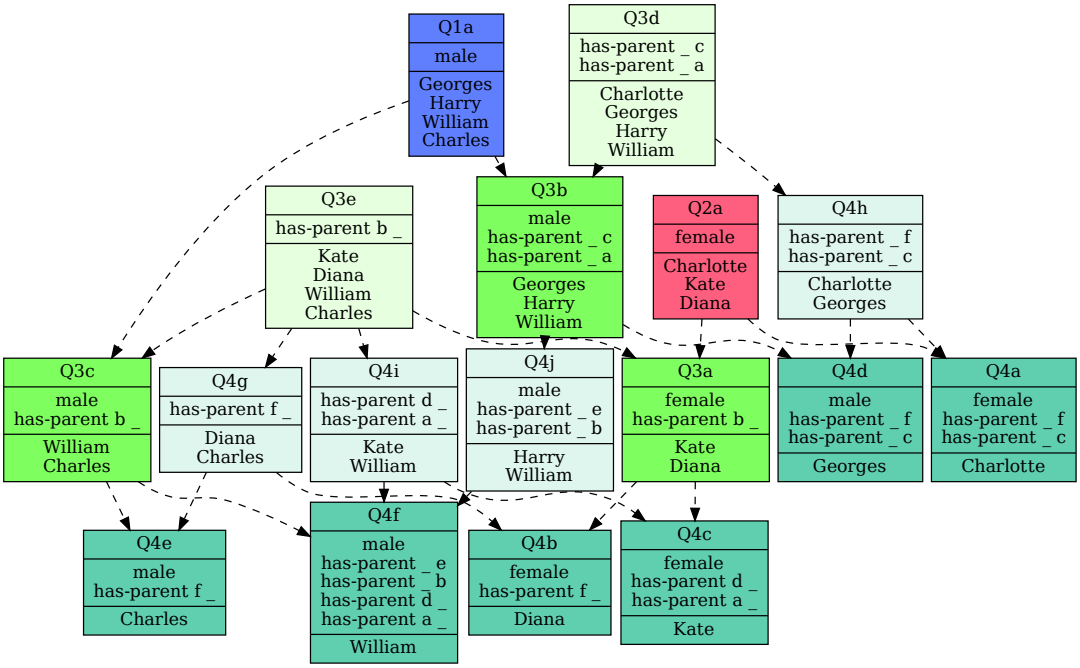

FIGURE 4.13 – Hiérarchie de concepts de l'ensemble de graph patterns de la figure 4.12.

4.6 Notion de concepts automorphiques

La duplication des concepts dans les résultats de GCA est parfois nécessaire pour représenter correctement les graph patterns qui capturent des symétries ou certaines structures dans le contexte graphe. Nous appelons ces concepts dupliqués *concepts automorphes*, qui sont des nœuds interchangeables dans un pattern. En effet, ils ont des intensions équivalentes et ont la même extension.

Pour illustrer cette notion de concepts automorphes, utilisons le contexte graphe illustré à la figure 4.15 qui décrit les relations entre les plats, les céréales et les pays. Ce contexte graphe est composé de deux sous-graphes : les composantes **C1** et **C2** (cycle de longueur 3). L'ensemble des patterns produits par GCA (résultant du produit $I \times I$) sans extension des concepts est présenté à la figure 4.16. Nous pouvons identifier que le pattern Q1 (en bleu) capture la composante **C2** du contexte graphe, et résulte du sous-produit de **C2** par lui-même (**C2** × **C2**). Il en est de même pour le pattern Q3 qui résulte du sous-produit de **C1** par lui-même et qui met en évidence un motif plus complexe et plus enrichit qui révèle les interactions croisées de **C1** × **C1**. Quant au pattern Q2 (en rouge), il capture la structure de la composante **C1** et généralise les composantes **C1** et **C2**, car résulte du sous-produit **C1** × **C2** ou (**C2** × **C1**, inversement).

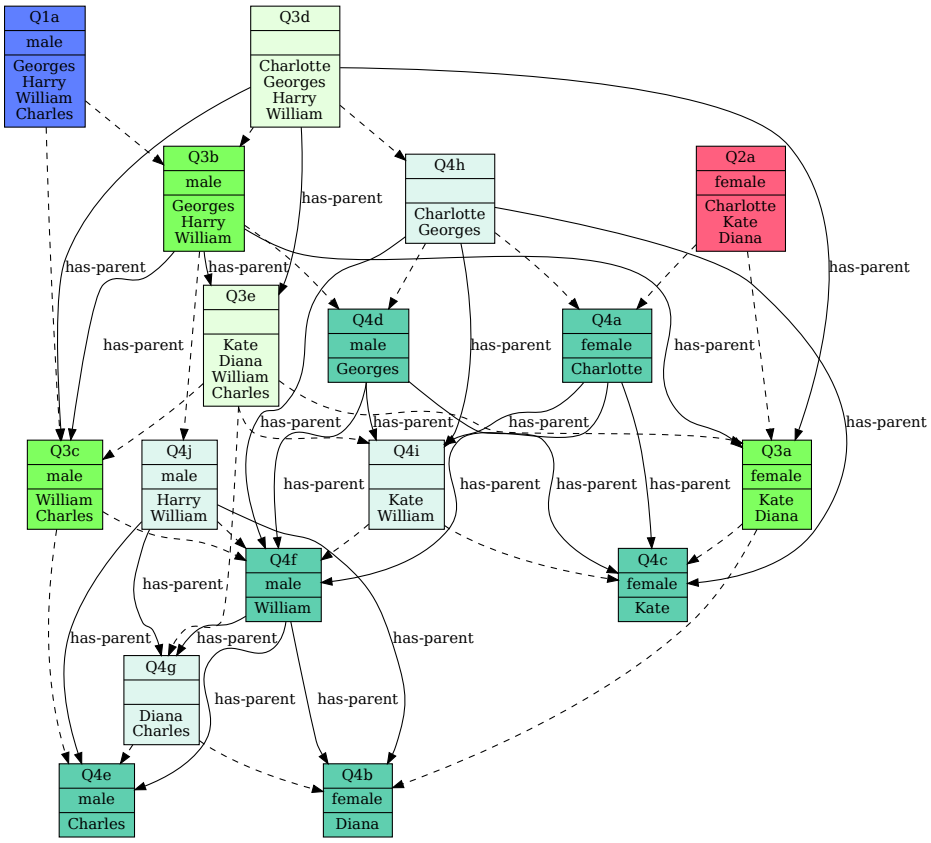

Figure 4.14 – Résultats de GCA en représentation combinée (figure 4.13 + figure 4.12).

En faisant un zoom sur le pattern de nœuds Q2 (figure 4.17), nous remarquons les nœuds $Q2a$ et $Q2f$ sont étiquetés (*dish*), $Q2c$ et $Q2e$ sont étiquetés $\{$*rice, cereal*$\}$ et pour terminer, $Q2b$ et $Q2d$ ont pour étiquette *country*. La figure 4.18 met en évidence le pattern de concepts Q2 dans lequel les extensions (instances des nœuds) sont calculées. Comme nous le constatons, ce pattern comporte six nœuds, mais seulement trois concepts unaires distincts (correspondant à trois extensions différentes). Les trois autres nœuds sont en réalité des duplications, ce qui justifie leur désignation en tant que concepts automorphes. Par exemple, les concepts $Q2a$ et $Q2f$ sont automorphes, car ont la même extension et une intension équivalente (les plats reliés à leurs principales céréales qui sont reliées à leur pays de production). Il ne s'agit donc que de différentes représentations d'un même concept théorique. Aussi, les concepts $Q2c$ et $Q2e$ sont automorphes au même titre que le sont $Q2b$ et $Q2d$.

Tout comme le traitement automatique des relations dans les deux sens, le phénomène de concepts automophes est intrinsèquement lié à la définition du produit catégorique de graphes qui capture les interactions combinées entre les motifs du contexte graphe et par

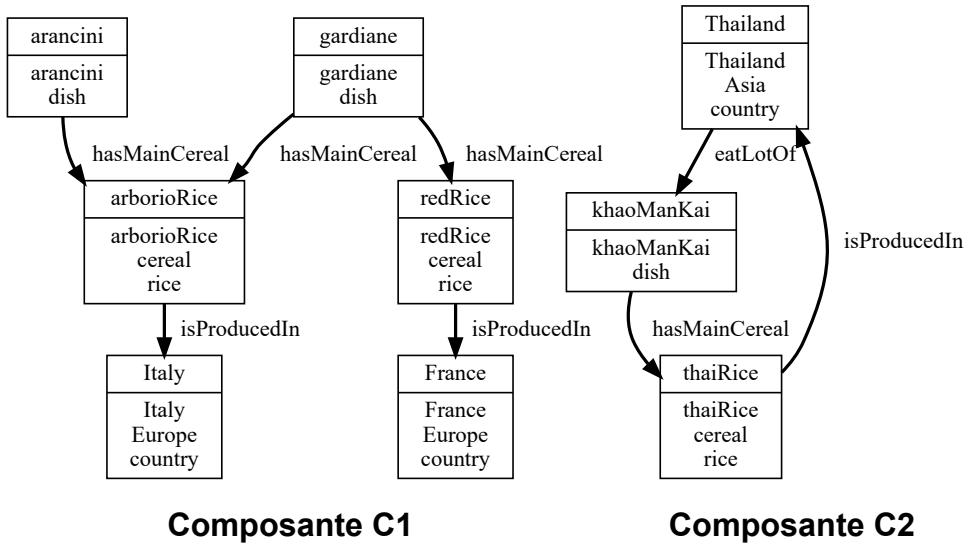

FIGURE 4.15 – Contexte graphe à propos des plats, des céréales et des pays (@M. Huchard).

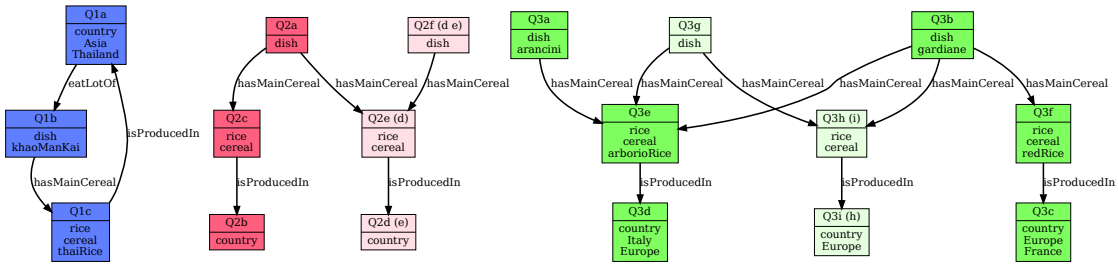

FIGURE 4.16 – Graph patterns (hors projection) du contexte graphe de la figure 4.15.

conséquent, capture indirectement les motifs du contexte graphe.

Pour une représentation hiérarchique qui respecte la structure de treillis, les concepts automorphes sont représentés dans la vue hiérarchique par des méta-nœuds (boîtes en pointillés) les regroupant. Par exemple, $Q2d$ et $Q2b$ dans la figure 4.19 qui fournit la représentation hiérarchique de l'ensemble des concepts graphes du contexte de la figure 4.15.

Il est important de remarquer la notation parenthésée dans les noms (identifiants) de certains concepts en l'occurrence : $Q2d\,(e)$, $Q2e\,(d)$ et $Q2f\,(d\ e)$. Les éléments entre parenthèses indiquent les nœuds utiles à la description du concept concerné. Nous avons précédemment défini l'intension d'un concept Qix d'un pattern P comme étant le PGP $((x), Pi_r)$, où Pi_r désigne le plus petit retract de P contenant le nœud x. Avec cette notation parenthésée, Pi_r devient le plus petit retract de P qui contient x *et les nœuds indiqués entre parenthèses*. Pour illustration, $int(Q2d\,(e)) = ((d), P2_r)$ où $P2_r$ est constitué de tous les

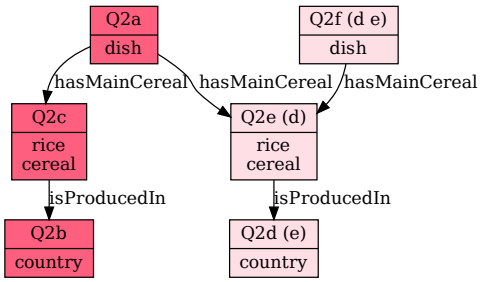

Figure 4.17 – Pattern de nœuds Q2.

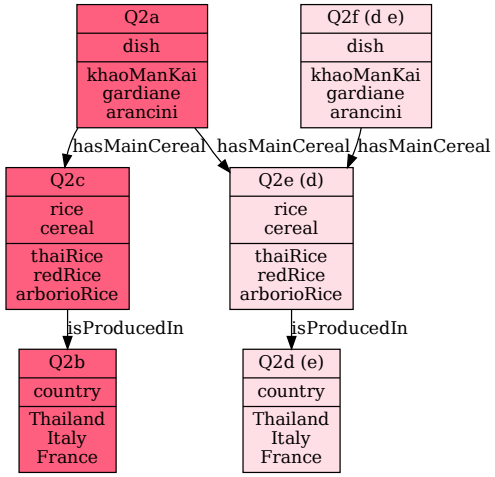

Figure 4.18 – Pattern de concepts Q2.

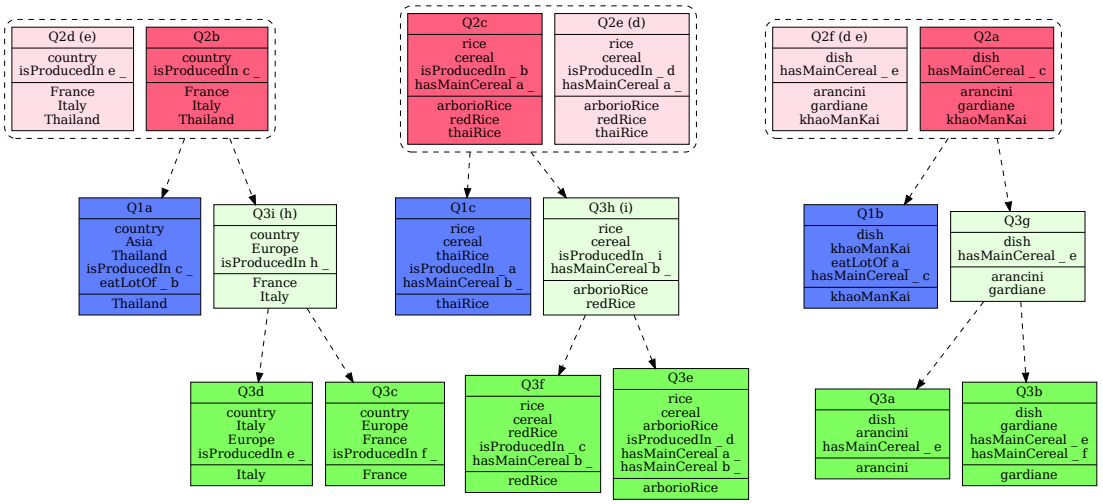

Figure 4.19 – Hiérarchie de concepts GCA sur le contexte graphe de la figure 4.15.

nœuds du pattern Q2 (figure 4.18) à l'exception du nœud $Q2f\ (d\ e)$. Par contre, l'intension de $Q2f\ (d\ e)$ impliquera tous les nœuds de Q2.

4.7 Conclusion

Ce chapitre a présenté GCA qui étend l'Analyse Formelle de Concepts (AFC) aux graphes, dont la structure permet une représentation flexible des données sous forme d'entités reliées par des relations d'arité quelconque. À partir d'un graphe de données appelé contexte graphe, GCA calcule les graph patterns qui mettent en évidence les différentes connexions entre les concepts, et par conséquent, différents schémas d'information sur l'ensemble du contexte graphe. Ces patterns sont calculés par produit catégorique de graphes et leurs instances sont obtenues par calcul de la projection de leurs nœuds sur le contexte graphe.

Notons que GCA a la capacité de traiter les relations $n(> 2)$-aires et que son opération fondamentale à savoir le produit catégorique de graphes lui confère le traitement automatique des relations dans les deux sens (direct et inverse), ainsi que la possibilité de capturer des symétries et certaines structures du contexte graphe, par le biais des concepts dits automorphes. En termes de présentation des résultats, GCA offre différentes possibilités en fonction de l'aspect principal visé par l'analyse : (1) la vue graph patterns pour mettre en évidence les structures relationnelles dans les données, (2) le vue hiérarchique pour mettre en évidence l'ordre de généralisation sur les concepts et (3) la vue combinée pour une représentation complète qui combine la vue graph patterns et la vue hiérarchique.

Le chapitre 5 suivant propose une synthèse des principaux travaux ayant exploré les liens ou rapprochements possibles entre RCA et GCA.

Chapitre 5

État de l'art sur les rapprochements entre RCA et GCA

Sommaire

5.1 Introduction

Comme présenté à la section 2.5, plusieurs extensions de l'Analyse Formelle de Concepts (AFC) [Ganter et Wille, 1999] ont été proposées pour le traitement de données complexes [Ferré, 2023 ; Ferré et Ridoux, 2000 ; Ganter et Kuznetsov, 2001] et de données multi-relationnelles [Bazin et al., 2024 ; Boffa, 2022 ; Kötters, 2013]. Parmi ces extensions, l'Analyse Relationnelle de Concepts (*Relational Concept Analysis* - RCA) [Huchard et al., 2007 ; Rouane-Hacene et al., 2013] et l'Analyse Conceptuelle des Graphes (Graph-FCA/GCA) [Ferré, 2015 ; Ferré et Cellier, 2020], toutes deux conçues pour les données multi-relationnelles, présentent quelques similitudes notamment : le calcul des concepts unaires, l'usage du quantificateur existentiel (∃), ainsi que le traitement des relations binaires.

En effet, le modèle de données de RCA, constitué d'une Famille Relationnelle de Contextes (FRC) représentant un ensemble de tables et leurs associations, peut être formalisé sous forme de graphe. Réciproquement, si l'on se limite aux relations binaires, le modèle de GCA qui est un contexte graphe, peut être aisément représenté par un ensemble de tables relationnelles. Ainsi, même du point de vue de la structure des données, RCA et GCA manipulent des représentations équivalentes lorsqu'on considère exclusivement des relations binaires.

Quelques travaux dans la littérature ont abordé, de manière directe ou indirecte, les relations entre les approches RCA et GCA, en se concentrant sur différents aspects, notamment :

— la lecture et l'interprétation de la famille de treillis produite par RCA, tâche reconnue comme difficile ;
— la question de la navigation au sein de la famille de treillis RCA, qui demeure centrale ;
— le traitement des relations n-aires par ces deux approches ;
— la comparaison de RCA et GCA sur des cas d'application concrets.

Ce chapitre propose une synthèse de ces travaux, regroupés selon les catégories de questions qu'ils soulèvent. La section 5.2 présente d'abord les aspects liés à la navigation, à l'exploration et à la visualisation des résultats produits par RCA. La section 5.3 examine ensuite les relations entre RCA et GCA sous l'angle de l'interprétation des ensembles de treillis relationnels, tandis que la section 5.4 traite des questions de modélisation et de comparaison entre les deux approches.

5.2 Naviguer, explorer et visualiser les résultats de RCA

RCA produit une famille de treillis, un pour chaque catégorie d'objets, reliés entre eux par des attributs relationnels qui traduisent les liens inter-objets initiaux. L'un des enjeux

majeurs pour une utilisation efficace de RCA réside ainsi dans l'interprétation de ces résultats. En effet, l'ensemble des treillis générés peut rapidement devenir difficile à interpréter, notamment lorsque leur taille est importante, car l'intension d'un concept peut dépendre récursivement de celles d'autres concepts. Pour remédier à cette complexité, plusieurs approches ont été proposées, allant de l'adaptation du processus d'exploration dans RCA au développement d'outils spécifiques dédiés à la visualisation et à la navigation dans les résultats. Cette section présente les principaux travaux menés dans ce cadre.

5.2.1 Adaptation de RCA pour une exploration progressive

Dolques et al. [Dolques et al., 2015] ont proposé une adaptation de RCA visant à rendre l'exploration des données progressive et interactive. Dans cette approche, l'utilisateur choisit, avant chaque itération de RCA, les contextes objets–attributs et objets–objets qu'il souhaite intégrer au processus d'analyse, ce qui permet de limiter le nombre de concepts générés et de rendre l'exploration plus flexible.

Plusieurs points de variation du processus RCA peuvent être exploités pour améliorer son utilisation dans un cadre d'analyse exploratoire. Pour chacun de ces points de variation, les auteurs proposent un *scénario alternatif* au processus classique de RCA (tel que décrit dans l'algorithme 3), dans lequel l'utilisateur est impliqué en choisissant l'étape suivante du déroulement. Les scénarios alternatifs proposés sont :

- *Initialisation* : construction des treillis uniquement pour les contextes objets–attributs sélectionnés, plutôt que pour l'ensemble des contextes. D'autres structures peuvent être construites, telles que les AOC-posets [Berry et al., 2012] ou les treillis Iceberg [Stumme et al., 2002].
- *Extension des contextes objets–attributs* : sélection d'un sous-ensemble de la famille de contextes relationnels et différents opérateurs de mise à l'échelle pour chaque contexte objets-objets sélectionné, au lieu de mettre à l'échelle tous les contextes objets-attributs à chaque étape. Il convient de noter qu'une relation objets-objets choisie nécessite un treillis de concepts regroupant les objets de son codomaine qui doivent avoir été calculés à une étape précédente.
- *Construction (mise à jour) des treillis* : construction uniquement des treillis correspondant aux contextes objets–attributs sélectionnés. D'autres structures peuvent également être construites à cette étape, telles que des AOC-posets ou des treillis Iceberg.
- *Arrêt du processus* : décision laissée à l'expert lorsque le point fixe n'est pas encore atteint.

Sur la base de ces variations possibles du processus RCA, les auteurs introduisent deux notions centrales pour formaliser l'exploration guidée des données : le chemin exploratoire (*exploratory path*) et le modèle de données par étapes (*step data-model*). Le modèle de données par étapes correspond à une configuration particulière du processus à une itération

donnée. Étant donnée une Famille Relationnelle de Contextes (FRC) $(\mathbf{K}, \mathbf{R})$, un *step data-model* DM_s (défini comme un triplet) décrit l'ensemble des choix de paramétrage effectués à une étape donnée : (1) la FRC partielle $(\mathbf{K}_s, \mathbf{R}_s)$ considérée, avec $\mathbf{K}_s \subseteq \mathbf{K}$ et $\mathbf{R}_s \subseteq \mathbf{R}$; (2) les quantificateurs de scaling associés à chaque contexte objets-objets de $\mathbf{R}_s$; (3) le choix de l'algorithme de construction des treillis (treillis complet, AOC-poset, treillis Iceberg avec un seuil donné, etc.) pour chaque contexte objets-attributs de $\mathbf{K}_s$.

Un chemin exploratoire est alors défini comme une séquence de modèles de données par étapes $(DM_{s0}, DM_{s1}, \ldots, DM_{sp})$, où chaque DM_{si} constitue la configuration utilisée à l'étape i du processus. Ce chemin obéit aux contraintes suivantes :

- le modèle de données de la première étape ne contient que des contextes objets-attributs ;
- à une étape $i \in \{1, 2, \ldots, p\}$, si un contexte objets-objets R_j fait partie du modèle de données, le contexte correspondant à son domaine doit également être inclus dans le modèle de données de cette même étape.
- à une étape $i \in \{1, 2, \ldots, p\}$, si un contexte objets-objets R_j fait partie du modèle de données, un treillis de concepts sur son codomaine devrait avoir été construit à une étape précédente $q < i$.

Cette approche d'exploration progressive a été implémentée dans l'outil RCAExplore [Dolques et al., 2019], qui permet à l'utilisateur de piloter le processus d'analyse en sélectionnant dynamiquement les contextes, quantificateurs et algorithmes à chaque étape. Ces adaptations du processus RCA classique visent à produire des résultats pertinents plus rapidement, en limitant le nombre de treillis construits (idéalement à ceux présentant un réel intérêt pour l'analyse) tout en réduisant la complexité globale de l'exploration et en offrant à l'expert la possibilité de guider la découverte selon son intuition et les motifs émergents au fil de l'analyse.

5.2.2 Réglage du processus d'exploration RCA

Pour aider l'analyste à anticiper, contrôler et interpréter les résultats produits par RCA, Ouzerdine et al. [Ouzerdine et al., 2022] proposent trois surcouches destinées à encadrer le processus d'analyse. En effet, RCA repose sur une grande variété d'opérateurs de mise à l'échelle, qui confèrent au processus d'analyse une forte flexibilité et une expressivité importante. Cependant, cette richesse peut également complexifier la tâche de l'analyste, en raison de la multiplicité des paramètres à configurer.

Pour répondre à cette difficulté, les auteurs intègrent ces sur-couches à l'outil RCAExplore [Dolques et al., 2019], qui permet une variété d'utilisations de RCA. Par exemple, dans RCAExplore, l'utilisateur peut à chaque étape du processus, changer les quantificateurs d'échelle, les contextes formels et les relations à considérer. Les trois sur-couches proposées sont présentées ci-après.

— La première sur-couche consiste à exprimer des contraintes sur les quantificateurs d'échelle, afin de garantir la cohérence de leur choix. En effet, RCAExplore offre la possibilité de choisir parmi plusieurs quantificateurs pour les relations, mais parfois, certaines relations sont sémantiquement liées et les quantificateurs qui leur sont associés doivent donc être cohérents. Les relations sont alors regroupées par classes d'équivalence de telle sorte que les relations d'une même classe reçoivent toutes le même quantificateur d'échelle à chaque étape du processus. Le quantificateur d'échelle d'une classe reste variable d'une étape à l'autre.
— La deuxième sur-couche consiste à faciliter l'interprétation des expressions des attributs, car l'une des difficultés rencontrées par l'utilisateur est de comprendre l'impact du choix des quantificateurs d'échelle sur son analyse. Pour cela, un interpréteur qui traduit automatiquement les choix réalisés sur l'interface utilisateur en une expression formatée respectant un langage fixé a été développé.
— La troisième sur-couche consiste à fournir des métriques quantitatives sur les treillis de concepts à construire et sur des règles d'implication relationnelle particulières afin d'aider les experts à affiner l'analyse. Les métriques permettent aux experts de disposer d'un aperçu sur les treillis (nombre de concepts, nombre et support des règles générées), et ainsi de réorienter l'analyse, en étendant ou en restreignant la recherche.

5.2.3 Navigation dans les treillis RCA

RCAviz[1] est un outil en ligne conçu pour faciliter la navigation et l'exploration au sein d'une famille de treillis issus de RCA [Huchard et al., 2024; Muller et al., 2022]. Il permet à l'utilisateur de définir un point de départ de la navigation en sélectionnant un sous-ensemble d'objets et d'attributs d'intérêt, servant de base à la navigation dans les treillis. L'outil prend en entrée les résultats de RCA (ensemble de treillis interconnectés) encodés au format JSON, lequel peut être généré à l'aide de l'outil FCA4J [Gutierrez et al., 2022].

Dans la pratique, la conception d'une visualisation interactive repose sur un processus itératif comprenant généralement trois étapes : (i) la définition des besoins et des structures de données adaptées, (ii) la conception d'encodages visuels et de mécanismes d'interaction répondant à ces besoins, et (iii) la validation auprès des utilisateurs, qui permet de réajuster les objectifs et les fonctionnalités au fil des itérations [Munzner, 2009; Sedlmair et al., 2012]. Dans le cas de RCAviz, les besoins fonctionnels et visuels ont été définis à l'issue de plusieurs réunions collaboratives réunissant trois experts en visualisation chargés du développement de la plateforme, un expert RCA, et un expert en données [Huchard et al., 2024]. La liste des besoins identifiés, formalisés et implémentés dans RCAviz, se décline en quatre points essentiels principaux.

1. *Sélection d'un concept.* Une navigation étape par étape nécessite un point de départ.

1. https://rcaviz.lirmm.fr/

Dans RCAviz, il s'agit d'un concept initial, obtenu après la sélection des attributs ou des objets qui intéressent l'utilisateur.

2. *Visualisation d'un concept et de ses voisins.* L'utilisateur doit pouvoir voir le contenu d'un concept (avec ses objets et ses attributs) ainsi que ses concepts voisins, c'est-à-dire les concepts de niveau inférieur (sous-concepts) et niveau supérieur (sur-concepts) dans le treillis et les concepts voisins dans les autres treillis.
3. *Navigation étape par étape.* L'utilisateur doit pouvoir explorer les concepts voisins du concept affiché (qu'ils appartiennent au même treillis ou à un autre).
4. *Historique.* L'utilisateur doit pouvoir conserver un historique de sa navigation, c'est-à-dire la liste des concepts précédemment explorés. Il doit également pouvoir naviguer dans cet historique et l'enregistrer afin de pouvoir reprendre son exploration ultérieurement.

Pour illustrer le fonctionnement de RCAviz, considérons de nouveau l'exemple de la FRC_1 ($\mathbf{K} = \{\mathcal{K}_{Person}, \mathcal{K}_{Car}, \mathcal{K}_{Garage}\}$, $\mathbf{R} = \{owner, sell, maintain\}$) présenté dans le tableau 3.1, à propos des personnes, des voitures et des garages.

La première étape du processus consiste à charger le résultat de RCA au format JSON. L'utilisateur peut ensuite soit conserver le contexte proposé par défaut (ici $\mathcal{K}_{Car}$), soit en sélectionner un autre ; dans notre exemple, le contexte choisi est $\mathcal{K}_{Garage}$. La figure 5.1 illustre l'étape de *sélection de concepts*, dans laquelle les garages A et B sont choisis comme objets d'intérêt, ainsi l'attribut relationnel $\exists maintain : 42$. Le *concept 38*, mis en évidence en gras, correspond au concept introduisant l'attribut relationnel sélectionné (avant la sélection de cet attribut, plusieurs concepts candidats étaient possibles : 5, 34, 38 et 39). Le concept 38 représente ainsi le concept de garages d'extension $\{A, B\}$. Une fois ce concept sélectionné, l'utilisateur valide la sélection via le bouton *Confirm* pour initier la navigation.

La figure 5.2 présente ensuite *l'interface de navigation* après cinq étapes d'exploration, débutant à partir du concept 38. Cette interface montre l'historique de navigation (panneau supérieur) retraçant la séquence des concepts explorés et l'explorateur (panneaux inférieurs). L'explorateur est divisé en trois sous-panneaux :

1. Panneau central : affiche le concept actuellement sélectionné (ici le concept 8), positionné au centre, ainsi que son *voisinage*, c'est-à-dire ses concepts parents et enfants dans le treillis du contexte courant ($\mathcal{K}_{Garage}$).
2. Panneau gauche : présente le contexte précédemment exploré, permettant de retracer le parcours de navigation.
3. Panneau droit : s'active lorsque l'utilisateur survole un lien dans le panneau central, affichant alors un aperçu du contexte cible de ce lien ; dans cet exemple, le *concept 24* de $\mathcal{K}_{Car}$.

Pour résumer, dans RCAviz, la navigation se compose d'une vue initiale permettant de sélectionner le concept de départ (le *sélecteur de concepts*) et de deux vues coordonnées : *l'explorateur* qui permet de naviguer étape par étape et *l'historique* qui permet de retracer le parcours de navigation. L'historique de navigation met en évidence les dépendances

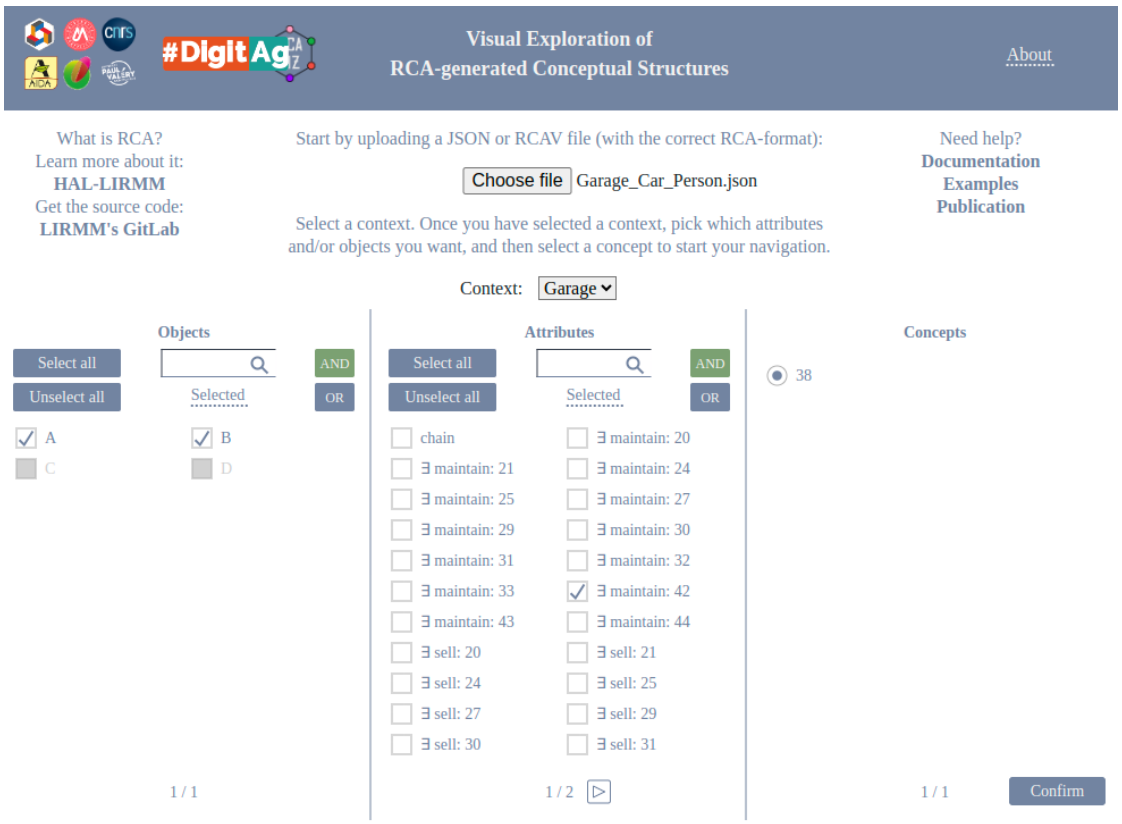

FIGURE 5.1 – Sélection du *concept 38* pour exploration

relationnelles entre les concepts explorés, ce qui s'apparente dans une certaine mesure à la vue graph pattern dans GCA qui met en évidence les structures relationnelles entre les concepts.

5.2.4 Navigation basée sur les requêtes

Naviguer dans une famille de treillis interconnectés afin de trouver des concepts d'intérêt n'est pas une tâche triviale, en raison de la taille potentiellement importante des treillis et de la nécessité pour l'expert de passer d'un treillis à l'autre. Azmeh et al. [AZMEH et al., 2011] proposent une approche de navigation dans une famille de treillis de concepts à partir des requêtes relationnelles formulées par un expert.

Une requête relationnelle est définie comme une composition de plusieurs requêtes simples[2] [MESSAI et al., 2005], enrichies de contraintes relationnelles. Par exemple, sur un contexte formel décrivant les pays selon leur continent, une requête simple pourrait être :

2. Le terme *requête simple* renvoie à une requête formulée à partir d'un seul contexte formel.

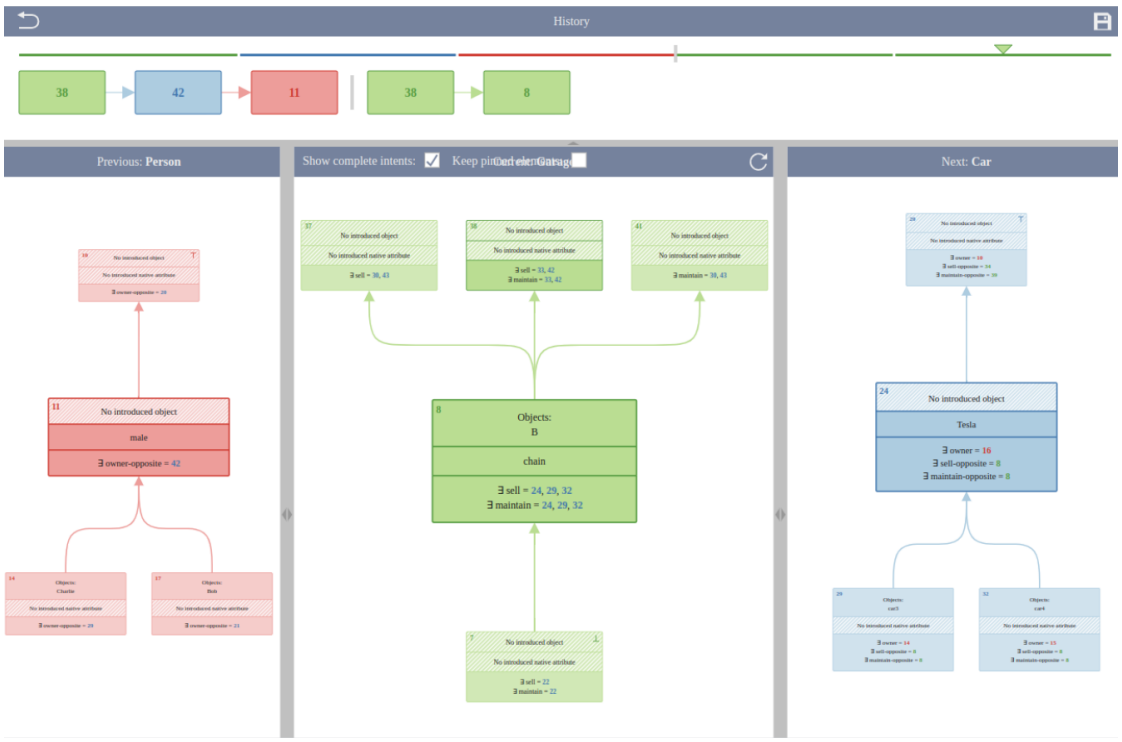

Figure 5.2 – Interface d'exploration RCAviz : historique (en haut), contexte précédent (à gauche), contexte actuel (au centre), contexte suivant (à droite).

quels sont les pays d'Europe ?

Par définition, une requête relationnelle est étroitement liée au modèle de données sous-jacent et doit respecter la structure de la Famille Relationnelle de Contextes (FRC) sur laquelle elle s'appuie. À titre d'illustration, considérons une FRC décrivant plusieurs entités (les pays, les restaurants, les plats mexicains, les ingrédients et les sauces (*salsas*)) ainsi que les relations qui les relient, comme représenté dans le schéma de la figure 5.3 (partie gauche). Un exemple de requête relationnelle q_r formulée sur cette FRC pourrait être le suivant : « Je recherche un pays ayant pour attribut "fr", qui *possède* un restaurant *servant* des plats *contenant* du poulet, du fromage et des tortillas de maïs ; je recherche également une salsa "piquante" qui *accompagne* ce plat. » [Azmeh et al., 2011]. Le schéma correspondant à cette requête reprend celui de la FRC initiale, à l'exception de la relation *made-in*, qui n'est pas utilisée dans la formulation de la requête q_r, comme illustré dans la partie droite de la figure 5.3.

De manière générale, à partir des données multi-relationnelles (FRC) et d'une requête qui contient des variables que l'on cherche à instancier, deux objectifs principaux sont poursuivis :

— Trouver l'ensemble des objets satisfaisant la requête, chaque réponse correspondant à une instanciation possible des variables présentes dans la requête ;

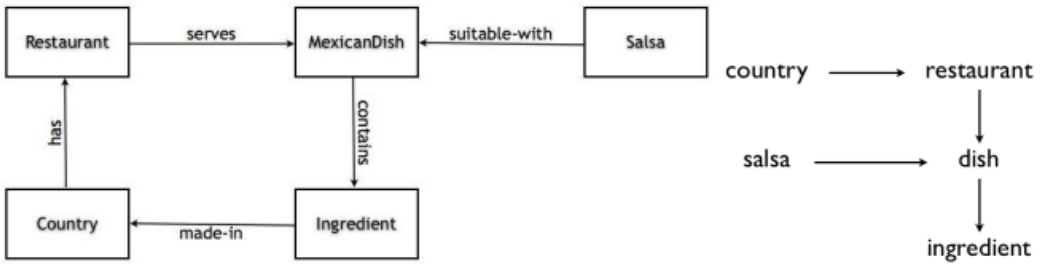

FIGURE 5.3 – Modèle données (à gauche) et schéma de requête (à droite) [AZMEH et al., 2011].

— Classer les objets associés à chaque variable selon leur degré de satisfaction des contraintes exprimées dans la requête, afin de dégager des réponses alternatives.

La requête relationnelle peut être modélisée sous la forme d'un graphe orienté acyclique (Directed Acyclic Graph – DAG), dans lequel certains nœuds sont étiquetés par des variables et d'autres par des objets. Dans cette représentation, les nœuds correspondent de manière approximative aux contextes formels (objets–attributs), et les arêtes traduisent les contextes relationnels (objets–objets). Un ordre total sur les arêtes du DAG est défini par un expert afin de guider la séquence de parcours. Sur cette base, l'algorithme de navigation procède en explorant successivement les différents treillis selon cet ordre, ce qui permet d'identifier les objets satisfaisant la requête ainsi que d'éventuelles réponses alternatives. La notion de réponses alternatives est à rapprocher de celle de concepts voisins dans RCAviz.

La formulation des requêtes sous forme DAG comportant des variables à instancier présente une forte analogie avec la notion de *Projected Graph Pattern* (PGP) utilisée dans GCA. Dans GCA, les projections effectuées sur les nœuds des graph patterns permettent d'identifier les instances répondant aux contraintes définies par le PGP. Il convient de noter qu'avec GCA, les PGPs peuvent inclure des cycles, contrairement au cas où les requêtes sont considérées comme des DAGs. Par comparaison à RCAviz, la navigation fondée sur une requête formulée en DAG offre une exploration plus ciblée et un voisinage plus spécifique. En effet, le chemin de navigation est explicitement déterminé par l'ordre total imposé sur les arêtes du DAG, ce qui guide de manière contrôlée la navigation.

Parallèlement à la question de navigation, une autre approche pour faciliter l'analyse des résultats de RCA consiste à résumer la famille de treillis en un ensemble ordonné de graphes. Cela permet d'obtenir une vue d'ensemble des informations contenues dans la famille de treillis, car les graphes mettent en évidence les différentes connexions entre les concepts. Nous présentons brièvement les deux stratégies principales de la littérature pour la transformation de l'ensemble des treillis RCA en graphes. Il s'agit d'une part de l'extraction des graphes orientés acycliques [NICA et al., 2016a ; NICA et al., 2020] - également appelés *Closed Partially Ordered Patterns* (*CPO-patterns*)[CASAS-GARRIGA, 2005] - et d'autre part de l'extraction des composantes fortement connexes [FERRÉ et CELLIER, 2018].

5.3 Interprétation de l'ensemble des treillis relationnels

5.3.1 Analyse des données séquentielles avec RCA

Nica et al. proposent un processus complet d'exploration de données temporelles ou séquentielles fondé sur RCA [Nica et al., 2016a ; Nica et al., 2020]. Leur approche consiste à calculer des CPO-patterns (*Closed Partially Ordered patterns*) [Casas-Garriga, 2005] à partir des résultats produits par RCA. L'objectif principal est de faciliter l'interprétation des résultats par les experts, en exploitant la structure hiérarchique des treillis pour mettre en évidence la relation de généralisation entre les motifs extraits. Cette organisation hiérarchique des CPO-patterns constitue un support d'analyse visuelle et offre une vue d'ensemble des régularités découvertes.

Le processus d'exploration proposé se décompose en deux étapes principales :

1. Application de RCA à la Famille Relationnelle de Contextes (FRC) qui encode les données séquentielles à analyser, afin d'obtenir une famille de treillis de concepts, un par contexte objets-attributs. Notons que les concepts dans ces treillis contiennent deux types d'attributs relationnels : (1) les *attributs relationnels qualitatifs* qui encodent les relations "simples" entre les objets/évènements (Par exemple, l'examen médical *détecte* un symptôme grave.) et (2) les *attributs relationnels temporels* qui encode la chronologie entre les évènements (Par exemple, le test viral *est_précédé_par* un examen médical.)
2. Parcours des concepts interconnectés issus des résultats de RCA afin d'extraire directement un CPO-pattern pour chaque concept d'un treillis principal choisi (*main lattice*). Ce treillis principal est le treillis qui contient les objets d'intérêt à analyser (par exemple les tests viraux). Les CPO-patterns obtenus sont automatiquement organisés grâce à l'ordre de généralisation qui existe entre les concepts associés. Cette hiérarchie de graphes facilite également le processus de réponse aux requêtes sur les entités du treillis principal.

Une approche naïve d'extraction de patterns séquentiels à partir des intensions de concepts prend en compte tous les attributs relationnels qualitatifs et temporels. Néanmoins, certaines propriétés des résultats de RCA peuvent être utilisées pour améliorer le processus d'extraction. Ces propriétés permettent de réduire les redondances en ne considérant que les attributs relationnels qualitatifs pointant sur les concepts les plus spécifiques (Propriété 1) et d'élaguer les attributs relationnels temporels qui peuvent être déduits par transitivité (Propriété 2). Notons que ce travail se concentre uniquement sur le scaling existentiel ($\exists$) et ne traite pas d'autres types de scaling.

Propriété 1. Soient C_1 et C_2 deux concepts tels que $C_1 \preceq C_2$. Si un concept C est tel que $\exists\, r(C_1) \in Int(C)$, alors nous avons également $\exists\, r(C_2) \in Int(C)$. Pour cette raison, $\exists\, r(C_2)$

est considéré comme redondant pour la description de C.

Propriété 2. Soit t un attribut relationnel temporel. Soient C, C_1 et C_2 trois concepts tels que $\{\exists t(C_1), \exists t(C_2)\} \subseteq Int(C)$, et $\exists t(C_2) \in Int(C_1)$. Alors $\exists t(C_2) \in Int(C)$ peut être déduit de $\exists t(C_1) \in Int(C)$ à partir de la transitivité de la relation t.

L'algorithme de construction de la hiérarchie de CPO-patterns prend en entrée une famille de treillis en indiquant le treillis principal et sa sortie est un treillis de structures de CPO-patterns, c'est-à-dire un treillis dans lequel les concepts principaux (du treillis principal) sont étendus avec les CPO-patterns correspondants. Pour chaque concept principal C_p, dont l'intension comporte au moins un attribut relationnel temporel, une liste de concepts adjacents est établie selon une approche en largeur, sur la base des Propriétés 1 et 2. Les concepts adjacents sont ensuite explorés en s'appuyant sur les attributs relationnels temporels de leurs intensions. En répétant ce processus pour chaque relation temporelle dans l'intension de C_p , on obtient un graphe que l'on peut appeler son graphe d'intension temporelle (que l'on note G_{C_p}), car il rassemble l'ensemble des concepts qui sont reliés à C_p par un chemin de relations temporelles. Pour chaque concept exploré, un sommet est dérivé et étiqueté avec un itemset (extension du concept) qui se caractérise par ses attributs formels (au sens de l'AFC) et ses attributs relationnels qualitatifs. L'étiquetage des sommets est décrit en détail dans [Nica et al., 2016a] et se base sur l'analyse des attributs relationnels qualitatifs, en fonction de la généralité ou de la spécificité du concept auquel ils renvoient.

Plus précisément, un CPO-pattern résume un ensemble de motifs séquentiels fermés qui coexistent dans les mêmes séquences analysées [Casas-Garriga, 2005]. En effet, chaque attribut relationnel temporel d'une intension d'un concept principal permet d'extraire au moins un pattern séquentiel. Un CPO-pattern peut s'apparenter à un PGP (Projected Graph Pattern) de GCA dans une certaine mesure; le point commun étant une représentation compacte de l'intension d'un concept (du treillis principal) sous forme d'un graphe. Plus précisément, si on suppose que les relations entre les concepts de GCA sont des relations temporelles, un CPO-pattern d'un concept C, correspondra dans GCA au PGP $((C), P_r)$ ou P_r est le sous-graphe contenant tous les concepts utiles à la description de C. La différence majeure se trouve alors dans le fait que la structure du CPO-pattern (graph direct acyclique) est ici déterminée uniquement par les relations temporelles. De plus, un CPO-pattern est calculé pour chaque concept, contrairement aux graph patterns GCA, qui capturent les factorisations entre PGPs, en ce sens qu'un graph pattern regroupe pour chacun de ses concepts tous les concepts utiles à sa description.

Notons que ce travail d'extraction de CPO-patterns, initialement appliqué à des données séquentielles classiques, a également été étendu à l'exploration de données séquentielles hétérogènes, c'est-à-dire constituées d'éléments de nature différente [Nica et al., 2018]. Cette extension met ainsi en évidence la capacité de l'approche à analyser des données séquentielles quelle que soit leur complexité.

5.3.2 Hiérarchie des graphes de concepts sur les résultats de RCA

Toujours dans cette optique de faciliter la lecture des résultats de RCA, Ferré et Cellier montrent comment les hiérarchies des graphes de concepts peuvent faciliter l'interprétation des treillis RCA [Ferré et Cellier, 2018]. Les auteurs proposent une représentation équivalente et non redondante d'une famille de treillis RCA sous la forme d'une hiérarchie de graphes de concepts. Dans cette hiérarchie, chaque concept appartient à un seul graphe de concepts et chaque graphe de concepts montre les relations entre plusieurs concepts.

L'idée sous-jacente au calcul des graphes de concepts repose sur l'identification des structures relationnelles comme des sous-ensembles de concepts interdépendants issus de différents treillis, puis à les utiliser comme éléments constitutifs de la représentation hiérarchique. Ces structures relationnelles sont définies comme les composantes fortement connexes (*Strongly Connected Components - SCC*)[Even, 2011] du *graphe de dépendance* entre les concepts. L'intuition derrière ce graphe de dépendance (Définition 5.1) est que l'intension d'un concept dépend de ses ancêtres dans le treillis (dépendances intra-treillis induites par la relation de subsomption), mais aussi des concepts cibles de ses attributs relationnels (dépendances inter-treillis).

Définition 5.1 (Graphe de dépendance). Soit **L** un ensemble de treillis de RCA. Le graphe de dépendance de **L** est le graphe orienté $G_{\mathbf{L}} = (V, E)$ où :

- V est l'ensemble de tous les concepts de tous les treillis dans **L**, à l'exception des bottom concepts,
- E est l'ensemble des dépendances entre concepts $C_1 \to C_2$, lorsque C_2 est un concept parent de C_1 ($C_1 \preceq C_2$), où C_1 est étiqueté par un attribut relationnel $\exists\, r(C_2)$, c'est-à-dire $\exists\, r(C_2) \in Int(C_1)$.

À partir de la définition du graphe de dépendance $G_{\mathbf{L}}$, un SCC de $G_{\mathbf{L}}$ est un ensemble maximal de concepts (provenant éventuellement de plusieurs treillis) où chaque concept a un chemin de dépendance vers tous les autres concepts du SCC pour la définition de son intension. Les SCCs sont ensuite utilisés pour définir les graphes de concepts (Définition 5.2).

Définition 5.2 (Graphe de concepts). Un graphe de concepts est le sous-graphe de la famille de treillis (enrichi des arêtes relationnelles) qui est induit par un SCC de $G_{\mathbf{L}}$. Il mélange donc des concepts provenant de plusieurs treillis, ainsi que les arêtes de treillis (relation de subsomption) et des arêtes relationnelles (attributs relationnels).

Par ailleurs, on sait que les SCCs d'un graphe forment un graphe orienté acyclique[3], où $SCC1 \to SCC2$ si un concept quelconque de $SCC1$ dépend d'un concept quelconque de $SCC2$, c'est-à-dire un concept de $SCC1$ est en relation avec un concept de $SCC2$. Ainsi,

3. Un graphe orienté acyclique dans lequel les SCCs sont "contractés" en super-nœuds.

les graphes de concepts d'un ensemble de treillis RCA peuvent donc être organisés en une hiérarchie de graphes de concepts, comme la hiérarchie de CPO-patterns.

Pour illustration, considérons l'exemple de la famille royale présenté au chapitre 4. Pour calculer les graphes de concepts RCA dans cet exemple, les auteurs ajoutent la relation inverse "*child*" aux données. Le résultat de RCA est un treillis de concepts similaire à la hiérarchie de concepts présentée à la figure 4.13, à la différence que le top concept y est explicitement représenté. La figure 5.4 illustre la hiérarchie des graphes de concepts obtenus pour cet exemple, telle que présentée dans [Ferré et Cellier, 2018]. Dans cette hiérarchie, les concepts sont regroupés en cinq graphes de concepts. Pour interpréter ces graphes, nous nous appuyons sur les résultats de GCA obtenus pour cet exemple, à savoir l'ensemble des graph patterns de la figure 4.12. En effet, les graph patterns de GCA sont équivalents aux graphes de concepts RCA obtenus pour cet exemple, à quelques différences de représentation près [Ferré et Cellier, 2018].

- Le graphe de concepts **G5** n'est pas détaillé pour des raisons de lisibilité ; il correspond au graph pattern Q4 (figure 4.12), constitué de dix concepts. Il contient des concepts les plus spécifiques, ce qui explique sa position au bas de la hiérarchie.
- Le graphe **G4** correspond au graph pattern Q3 (figure 4.12), où, par exemple, le nœud $p10$ représente le concept des parents et correspond au concept $Q3d$.
- Les graphes **G3** et **G2** représentent respectivement les concepts des personnes de caractéristique *female* (concept $Q2a$) et *male* (concept $Q1a$)
- Enfin, **G1** représente le concept englobant toutes les personnes, c'est-à-dire le top concept.

Les auteurs soulignent toutefois que l'équivalence observée entre RCA et GCA dans cet exemple ne saurait être généralisée. En effet, cette correspondance ne se vérifie pas dans d'autres cas étudiés, notamment celui portant sur les relations entre patients et traitements présenté dans [Rouane-Hacene et al., 2013].

En comparaison au travail précédent sur l'extraction des CPO-patterns [Nica et al., 2016a ; Nica et al., 2020], ce travail d'extraction des SCCs ne considère également que la mise à l'échelle existentielle ($\exists$), bien que l'approche reste applicable à d'autres opérateurs de mise à l'échelle. En complément de l'objectif d'interprétation visé par ces deux travaux, voici quelques points de divergence :

- Les CPO-patterns sont définis pour les données séquentielles et sont uniquement générés pour les concepts d'un treillis choisi (*main lattice*). La génération des CPO-patterns pour chaque concept du treillis choisi peut entraîner l'omission de certaines factorisations potentielles entre les patterns, et ainsi faire perdre des informations intéressantes.
- Le calcul des graphes de concepts par extraction des composantes fortement connexes ne fait aucune hypothèse sur la famille de contextes, et peut en conséquence traiter

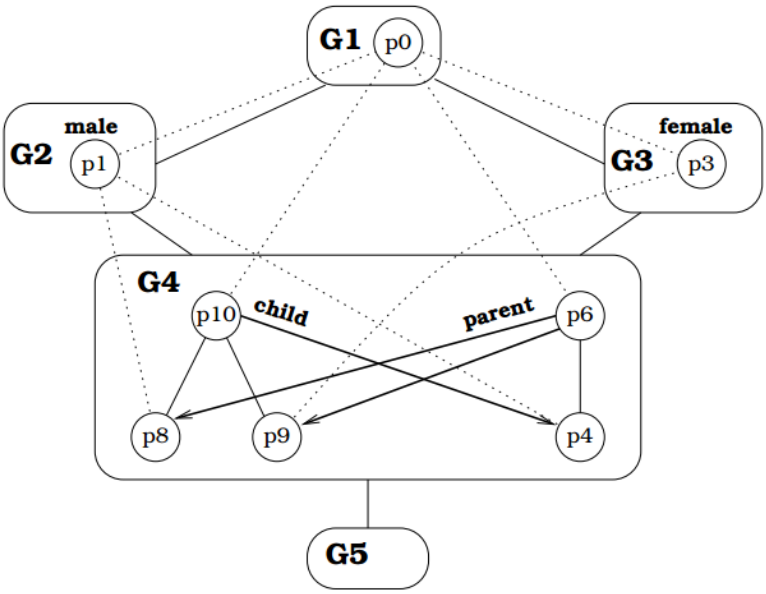

Figure 5.4 – Hiérarchie des graphes de concepts pour l'exemple de la famille royale : les flèches indiquent les relations entre concepts, les lignes pointillées représentent la relation de subsomption entre les concepts [Ferré et Cellier, 2018].

toutes sortes de structures relationnelles et pas seulement des séquences. Il n'est pas nécessaire de choisir un treillis comme point de départ comme avec les CPO-patterns.

En comparaison à GCA, un graphe de concepts est similaire à un graph pattern GCA à la différence que la relation de subsomption entre les concepts n'est pas représentée dans les patterns GCA. En effet, un graphe de concepts met en évidence les différentes relations entre les concepts et chaque concept a un chemin de dépendance vers tous les autres concepts du SCC. Dans un graph pattern GCA, tous les concepts interdépendants, du point de vue de la définition des intensions sont membre d'un seul et même pattern. En outre, cette représentation hiérarchique des graphes de concepts se rapproche de la représentation combinée des résultats dans GCA qui combine justement les représentations graphique (vue graph patterns) et hiérarchique. Tandis qu'en GCA, la représentation combinée peut complexifier la lecture, la hiérarchie des graphes de concepts privilégie un ordre de généralisation porté sur les graphes, tout en maintenant la subsomption entre concepts, ce qui facilite la visualisation.

Comme nous pouvons le constater, l'ensemble de ces travaux ne comparent pas directement RCA et GCA, ni en termes de fonctionnement, ni en termes de résultats. Toutefois, le point commun entre ces travaux réside dans la transformation des résultats de RCA sous forme de graphes, ainsi que dans la navigation à travers les treillis, qui impliquent de traverser le graphe de dépendance entre les concepts. Cela évoque immédiatement GCA, qui

calcule les graph patterns, et montre qu'il est possible de passer de la structure de pattern à une structure hiérarchique en application un ordre de généralisation sur l'ensemble des concepts des graph patterns. À cet stade, des similitudes potentielles entre RCA et GCA peuvent être envisagées, d'autant plus que le passage des treillis RCA aux graphes rejoint l'objectif des graph patterns GCA : faciliter l'interprétation tout en capturant les régularités présentes dans les données ; ce qui contribue également au traitement des requêtes.

5.4 Modélisation et comparaison

Les travaux présentés jusqu'ici font un rapprochement indirect entre RCA et GCA grâce à la transformation des résultats de RCA sous forme de graphes. Keip et al. ont effectué une comparaison pratique (sur un jeu de données réel) de TCA (*Triadic Concept Analysis*) [Lehmann et Wille, 1995], RCA et GCA pour la modélisation des valeurs indéterminées dans les données ternaires [Keip et al., 2020]. Dans ce travail, les relations ternaires sont directement traitées par les approches TCA et GCA, tandis que RCA nécessite d'effectuer des transformations en amont, pour encoder ces relations ternaires au format binaire.

En réalité, la représentation des données relationnelles est une question récurrente dans de nombreux domaines, de la fouille de données [Džeroski, 2003], des bases de données [Jones et Song, 2000 ; Song et Jones, 1995] aux Logiques de Descriptions (LD) [Baader et al., 2003]. Par exemple [Hodo et al., 2023] combine FCA et LD pour calculer les clusters sur les graphes de connaissances, de manière similaire à RCA. Pour ce qui concerne RCA, Keip et al. ont utilisé RCA pour le traitement des relations ternaires en appliquant certaines transformations et encodages aux données sources [Keip et al., 2019].

5.4.1 Modélisation des données ternaires en RCA

Au-delà des relations binaires, les relations d'arité élevée doivent être transformées pour être traitées par RCA. L'effet de la formulation des données dans RCA pour un modèle de données contenant plusieurs relations binaires et une relation ternaire a été étudié par [Keip et al., 2019]. Dans ce travail, deux modèles d'encodage à savoir la *réification* et la *décomposition* sont utilisés pour transformer la relation ternaire en relations binaires.

— *Réification.* La réification d'une relation ternaire r a consisté à représenter r en plusieurs relations binaires. Cette représentation passe par l'ajout d'un contexte objets-attributs intermédiaire dont les entités sont les triplets d'objets, instances de r et qui sont reliés aux entités de départ par le biais des contextes objets-objets (relations binaires). Formellement, une relation ternaire $r(A, B, C)$ est transformée en 3 relations binaires : $r1(ABC, A)$, $r2(ABC, B)$ et $r3(ABC, C)$ où ABC est le contexte objets-attributs contenant les instances de r c'est-à-dire des triplets d'objets.

— *Décomposition.* Dans ce travail, la décomposition de la relation ternaire r a consisté

à projeter r en trois relations binaires, une pour chaque paire d'ensembles d'objets liés. La décomposition de la relation $r(A, B, C)$ s'encode en trois relations binaires : $r1(A, B)$, $r2(B, C)$, and $r3(A, C)$.

Les auteurs ont effectué une évaluation à la fois qualitative et quantitative de ces deux encodages sur un exemple concret. L'évaluation quantitative a consisté à examiner la taille des structures conceptuelles (ensemble de treillis interconnectés) produites. D'un point de vue global, l'encodage par réification a produit moins de concepts (et d'attributs relationnels) que la méthode par décomposition. Pour ce qui est de l'évaluation qualitative, elle a consisté en l'évaluation de leurs capacités respectives à répondre à un scénario de remplacement concernant la protection des plantes contre les ravageurs. En résumé, ces deux modèles d'encodage ont permis de satisfaire la requête présentée par l'expert du domaine. Cependant, un ensemble de trois relations binaires (encodage par décomposition) n'est pas équivalent à une relation ternaire. Ainsi, l'encodage par décomposition sera donc parfois moins précis, car il ne dispose pas de relation ternaire.

5.4.2 Comparaison Pratique de TCA, RCA et GCA

Comme évoqué précédemment, dans le travail de [Keip et al., 2020] qui propose une comparaison pratique de TCA, RCA et GCA sur un modèle de données réel constitué d'une relation seule ternaire, la relation ternaire a directement été traitée par TCA et GCA. En effet, GCA étend FCA aux relations n-aires et une relation ternaire peut être intégrée dans un schéma TCA, c'est-à-dire " un objet o possède l'attribut a sous la condition b ". Pour ce qui concerne RCA, des modélisations doivent être faites en amont pour transformer les relations ternaires en relations binaires compatibles au format RCA [Keip et al., 2019].

Pour cette application, trois différents encodages ont été utilisés pour pour transformer la relation ternaire.

- Le premier encodage a consisté en la réification d'une relation ternaire $r(A, B, C)$ en un ensemble de trois relations $r1(ABC, A)$, $r2(ABC, B)$ et $r3(ABC, C)$ comme décrit précédemment.
- Le deuxième encodage a consisté en une décomposition d'une relation $r(A, B, C)$ en une chaîne de deux relations binaires $r1(A, B)$ and $r2(B, C)$.
- Le troisième encodage (*partitionnement*) a consisté à partitionner une relation ternaire $r(A, B, C)$ suivant les entités de la catégorie B en un ensemble $\{b_i(A, C)\}_{b_i \in B}$ de relations binaires, de sorte que chaque instance b_i de B devient une relation. Avec cet encodage, le treillis de concepts de la catégorie B est perdu.

Les auteurs comparent les trois approches (TCA, RCA et GCA) sur leurs concepts, leur flexibilité, leur lisibilité et leur facilité d'utilisation. En ce qui concerne la lisibilité et la facilité d'utilisation, TCA et GCA présentent tous deux leurs résultats sous la forme d'un ensemble unique. TCA énumère les concepts triadiques, tandis que GCA relie ses concepts dans des graph patterns. RCA relie également les concepts, mais il est nécessaire de passer

d'un treillis à l'autre pour accéder à la classification globale. Un encodage de la relation ternaire en relations binaires a été utilisé avec GCA, pour illustrer la facilité d'interprétation des résultats de GCA, sur les relations binaires par rapport à ses résultats sur les relations ternaires. Pour ce qui est de la visibilité des hiérarchies de concepts et des structures relationnelles, les treillis de concepts triadiques sont difficiles à visualiser et à comprendre dans leur intégralité, même avec le paradigme de projection utilisé par l'outil FCA *Tools Bundle* [Kis et al., 2016]. RCA représente clairement les hiérarchies de concepts, une pour chaque type d'objet, mais pas les patterns/structures relationnels comme GCA. GCA dispose de trois modes de sortie : affichage des hiérarchies uniquement, comme RCA, affichage des modèles relationnels uniquement, ou affichage des deux combinés. Pour cette application, TCA est l'approche qui produit le moins de concepts, suivie de RCA avec *partitionnement*, puis RCA avec *décomposition en chaîne*, GCA et enfin RCA avec *réification*, qui est celle produisant le plus grand nombre de concepts.

Pour terminer, notons que la transformation des données contenant des relations ternaires en une famille relationnelle de contextes (binaires) n'est pas une tâche triviale et peut souvent conduire à des pertes d'informations suivant le type d'encodage utilisé. C'est dans cette dynamique qu'une piste d'extension de RCA [Rouane-Hacene et al., 2013] pour prendre en compte les relations ternaires a été explorée dans [Leutwyler et al., 2022]. Les auteurs définissent une famille relationnelle de contextes ternaires comme étant une paire (K, R) où K est un ensemble de contextes formels et R est un ensemble de relations binaires ou ternaires entre des objets, dont au moins une est ternaire entre les concepts. Un attribut relationnel ternaire devient un attribut relationnel qui pointe sur deux concepts cibles de sorte à former une arête ternaire. Cette extension de RCA aux relations ternaires reste limitée au quantificateur existentiel ($\exists$) et mérite d'être évaluée dans différents scénarios afin d'analyser sa scalabilité, sa complexité et la lisibilité des résultats dans un contexte d'application pratique.

5.5 Conclusion

Dans ce chapitre, nous avons présenté une synthèse des principaux travaux de la littérature traitant, de manière directe ou indirecte, des liens existant entre les deux approches, RCA et GCA.

Comme nous avons pu le constater, les travaux sur ce sujet restent relativement peu nombreux. Certains établissent un rapprochement indirect entre RCA et GCA en transformant les résultats de RCA sous forme de graphes afin de faciliter leur interprétation. L'étude de [Keip et al., 2020] présente une comparaison pratique entre RCA et GCA sur un jeu de données réel, limitée à l'analyse des résultats en termes de flexibilité et de lisibilité. Ces travaux suggèrent l'existence de similitudes potentielles entre les deux approches. Toutefois, les comparaisons réalisées jusqu'à présent entre RCA et GCA se sont principalement concentrées sur des aspects spécifiques, tels que l'interprétation des résultats, et

aucune étude globale n'a encore analysé de manière approfondie leurs caractéristiques ni les résultats qu'elles produisent.

Dans la deuxième partie de cette thèse, nous présentons nos contributions, qui portent sur une comparaison à la fois empirique et théorique des deux approches RCA et GCA. Le chapitre 6 introduit la méthodologique adoptée pour conduire cette étude comparative.

Deuxième partie

Contributions

Chapitre 6

Approche Méthodologique

Comme présenté dans les chapitres précédents, l'Analyse Relationnelle de Concepts (*Relational Concept Analysis* - RCA) [Rouane-Hacene et al., 2013] et l'analyse conceptuelle des graphes (Graph-FCA/GCA) [Ferré et Cellier, 2020] sont deux extensions de l'Analyse Formelle de Concepts (AFC) [Ganter et Wille, 1999] pour le traitement des données multi-relationnelles. Bien que leurs objectifs et leurs résultats semblent similaires, la modélisation des données et la définition des concepts sont différentes dans les deux approches.

Les rapprochements réalisés jusqu'à présent entre RCA et GCA se sont focalisés sur certains aspects spécifiques, tels que l'interprétation et la visualisation des résultats, comme exposé au chapitre 5. Il apparaît donc pertinent d'élargir la comparaison entre RCA et GCA, de manière à fournir à l'analyste des éléments d'aide au choix de l'approche la plus adaptée, selon la nature des données ou des résultats attendus. En termes de résultats, le concept constitue l'unité de connaissance commune aux deux approches. Ainsi, la question centrale de leur comparaison consiste à déterminer *comment l'ensemble des concepts RCA se situe par rapport à l'ensemble de concepts GCA, et réciproquement*. Répondre à une telle question suppose naturellement d'examiner le positionnement d'un concept RCA par rapport à un concept GCA, et inversement.

Un concept est constitué de deux dimensions : **l'extension** qui représente les objets (instances) du concept, et **l'intension** qui décrit l'ensemble des caractéristiques que partagent ces objets. RCA définit les extensions de concepts comme des ensembles d'objets et les intensions de concepts par des ensembles d'attributs formels et relationnels. GCA définit les extensions de concepts comme des ensembles de tuples d'objets et les intensions comme des *Projected Graph Patterns* (PGP). Pour comparer un concept RCA à un concept GCA, il convient dans un premier temps d'examiner leurs extensions, puis, dans un second temps, d'analyser leurs intensions, car ces deux dimensions sont définies différemment dans les deux approches.

Bien que RCA et GCA visent toutes deux à calculer des concepts sur des données multi-relationnelles tout en capturant les relations entre les objets, elles diffèrent sur de nombreux aspects. Nous avons donc structuré notre étude comparative selon deux axes principaux : la comparaison dans leur cadre commun et celle sur leurs différences, comme l'illustre la figure 6.1. Le premier axe, relatif à leur paramétrage commun, se décline en deux dimensions : la dimension extensionnelle et la dimension intensionnelle.

Nous débutons au chapitre 7 avec la comparaison extensionnelle des deux approches et

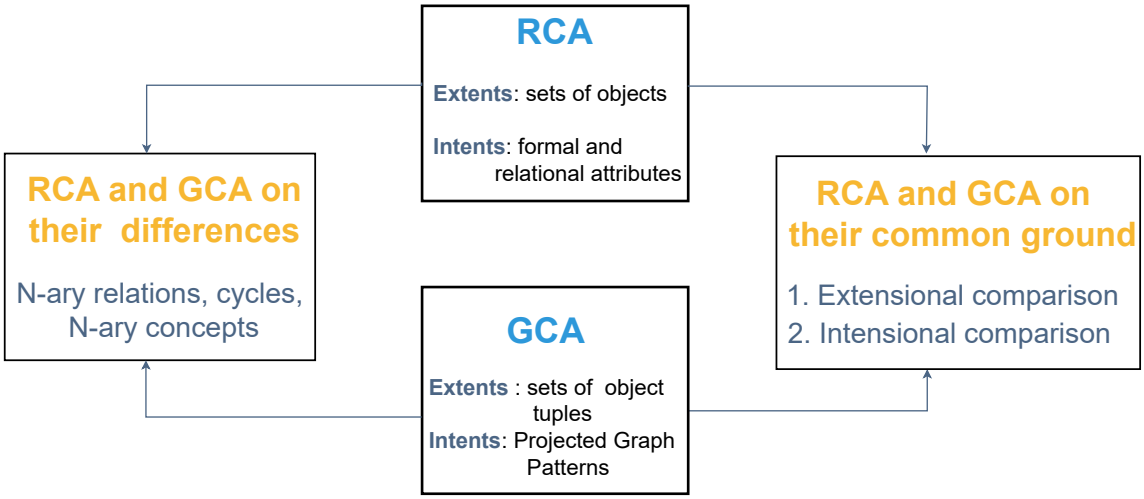

Figure 6.1 – Axes de comparaison examinés entre RCA et GCA.

montrons que l'ensemble des extensions de concepts RCA est inclus dans l'ensemble des extensions de concepts GCA. Autrement dit, ce chapitre met en évidence que, sur un même jeu de données, à chaque concept RCA correspond un concept GCA possédant la même extension. En effet, les extensions des concepts RCA et GCA se définissent toutes deux comme des ensembles d'objets dans leur cadre commun. Après cette comparaison extensionnelle de RCA et GCA, nous poursuivons au chapitre 8 avec l'analyse intensionnelle, qui vise à caractériser la nature des relations qui existent entre l'intension d'un concept RCA et celle d'un concept GCA.

RCA et GCA définissent les intensions de concepts de manière distincte. Par conséquent, celles-ci ne sont pas directement comparables et nécessitent l'adoption d'une représentation commune. Nous avons retenu les graphes comme représentation partagée, car ils constituent une structure naturelle pour modéliser des données multi-relationnelles et parce que des travaux antérieurs vont dans ce sens pour RCA [Ferré et Cellier, 2018 ; Nica et al., 2016a]. La première étape a ainsi consisté à transformer la famille de treillis de concepts RCA en un ensemble de patterns relationnels comparables aux graph patterns GCA. Nous avons ensuite démontré que l'ensemble des intensions de concepts RCA est inclus dans celui des intensions de concepts GCA, en établissant que l'ensemble des patterns relationnels de RCA est inclus dans l'ensemble des graph patterns GCA. Combiné au résultat de la comparaison extensionnelle, ce résultat de la comparaison intensionnelle permet de conclure que l'ensemble des concepts RCA est inclus dans l'ensemble des concepts GCA.

Le deuxième axe de comparaison, centré sur les différences entre RCA et GCA, est exploré au chapitre 9. L'objectif est ici de comparer les deux approches sous un angle pratique, afin d'examiner dans quelle mesure leurs différences peuvent être complémentaires

et bénéfiques pour l'analyse. Ce chapitre aborde en particulier les questions relatives à la modélisation des relations n-aires et au traitement des cycles dans les deux approches.

En somme, la première contribution (chapitre 7) est consacrée à la **comparaison extensionnelle** de RCA et GCA dans leur cadre commun. La deuxième contribution (chapitre 8) porte sur la **comparaison intensionnelle** de ces deux approches, également dans leur cadre commun. Enfin, la troisième contribution (chapitre 9) propose une analyse comparative de RCA et GCA **du point de vue de leurs différences**.

Chapitre 7

Comparaison extensionnelle de RCA et GCA dans leur cadre commun

Sommaire

Ce chapitre présente notre première contribution sur la comparaison extensionnelle de RCA et GCA, tant de manière empirique que théorique. On montre que les extensions de concepts RCA sont incluses dans les extensions de concepts GCA. La section 7.1 dresse un aperçu général des différences entre RCA et GCA, et la section 7.2 aborde la modélisation des relations dans ces deux approches. La comparaison des extensions des concepts RCA et GCA est ensuite menée en section 7.3, à partir d'expérimentations sur des exemples. Enfin, la section 7.4 propose une validation théorique des observations faites sur les exemples, par le biais de démonstrations formelles.

7.1 Aperçu des différences entre de RCA et GCA

D'un point de vue général, RCA et GCA se distinguent à plusieurs niveaux, allant des principes algorithmiques, de la présentation des résultats aux types de données et relations pris en compte par chaque approche. Cette section fournit une brève description de l'ensemble des différences entre RCA et GCA.

Quantificateurs. GCA utilise (implicitement) uniquement le quantificateur existentiel $\exists$, tandis que RCA utilise une diversité de quantificateurs ($\exists$, $\exists\forall$, $\exists\forall_{\geq n\%}$, etc.) [Braud et al., 2018], qui confèrent au processus d'analyse une forte flexibilité en termes de précision de l'information qui peut être extraite. Pour illustration, sur un jeu de données à propos des repas et leurs ingrédients (avec une relation "*contient*" entre les repas et les ingrédients), on pourrait être intéressé par la recherche des repas dont au moins *50%* des ingrédients sont de même type ou ont les mêmes caractéristiques. En d'autres termes, on cherche des repas ayant au moins *50%* de leurs ingrédients dans un même concept des ingrédients. Par exemple, les repas dont *50%* des ingrédients sont des *légumes* (*légume* ici est un attribut des ingrédients). Il suffira pour filtrer, d'appliquer un scaling relationnel avec l'opérateur universal-percent $\exists\forall_{\geq 50\%}$ sur la relation *contient*. Ainsi, un attribut relationnel $\exists\forall_{\geq 50\%}$*contient*$(C\_ingredient)$ sera ajouté à un *repas* si et et seulement $C\_ingredient$ contient au moins *50%* de ses ingrédients. L'impact des quantificateurs a été étudié en détail dans [Braud et al., 2018].

Présentation des résultats. Le résultat de RCA consiste en une famille de treillis interconnectés par des attributs relationnels. Une difficulté réside dans la navigation entre les treillis lorsque l'ensemble de données est volumineux, c'est pourquoi des outils ont été proposés pour résumer ou simplifier l'interprétation des résultats de RCA [Ferré et Cellier, 2018 ; Huchard et al., 2024 ; Nica et al., 2016a]. Du côté de GCA, l'outil *gfca* [Ferré, 2019] dispose de trois modes de présentation des résultats : des hiérarchies de concepts comme la famille de treillis RCA, des structures relationnelles entre les concepts (graph patterns), et enfin une représentation combinée des hiérarchies de concepts et des graph patterns.

Processus algorithmique. D'un point de vue algorithmique, les concepts RCA sont calculés par application de l'AFC sur les contextes objets-attributs au fil des itérations de scaling jusqu'au point fixe, tandis que GCA opère par intersection de graphes jusqu'à saturation. Cette intersection de graphes, qui correspond au produit catégorique de graphes, permet à GCA de prendre automatiquement en compte les relations inverses. Les approches RCA et GCA disposent chacune d'un ensemble d'options permettant de contrôler les résultats. Par exemple, RCA peut recourir à différents algorithmes, tels que Ares [Dicky et al., 1994] et Hermes [Berry et al., 2012] pour le calcul des sous-hiérarchies de Galois [Godin et Mili, 1993], ou encore iceberg [Stumme et al., 2002], destiné au calcul du sup-demi-treillis. Par défaut, tous les concepts sont calculés. En ce qui concerne GCA, il existe plusieurs options

telles que : *-only-cores* pour limiter la génération aux *pattern cores*, *-minsupp N* pour limiter la sortie aux concepts unaires dont la taille de l'extension est supérieure ou égale à N.

Traitement des relations n-aires. RCA se limite au traitement des relations binaires, tandis que GCA est conçu pour traiter directement les relations d'arité quelconque. Par conséquent, le traitement des données contenant les relations d'arité supérieure à 2 nécessite des modélisations supplémentaires, pour les mettre au format RCA [Keip et al., 2020, 2019]. Du côté de GCA, la question peut se poser sur la lecture et l'interprétation des graph patterns lorsque les relations ont une grande arité.

Concepts n-aires. RCA se limite au calcul de concepts unaires, contrairement à GCA, qui calcule également des concepts $n(> 1)$-aires (appelés concepts de relations n-aires). Les extensions des concepts n-aires sont des ensembles de n-uplets d'objets et les intensions sont des *Projected Graph Patterns -PGP* dont les tuples de projection sont de longueur n (projection sur n éléments). De la même manière que les concepts unaires représentent des groupes d'éléments ayant des caractéristiques communes, les concepts n-aires représentent des groupes de tuples d'objets ayant des caractéristiques communes, et permettent ainsi d'exprimer des structures plus riches. Par exemple, dans un contexte graphe présentant les membres d'une famille avec une relation "*parent*", le concept binaire de "frère ou sœur" peut être découvert et décrit par un PGP comme "une paire de personnes ayant le même père (parent masculin) et la même mère (parent féminin)".

Définition des concepts. RCA et GCA définissent les concepts de deux manières différentes. RCA définit ses extensions de concepts par des ensembles d'objets et ses intensions par des ensembles d'attributs (formels et relationnels). Du côté de GCA, les extensions de concepts sont définies par des ensembles de tuples d'objets, et les intensions de concepts sont définies par des PGPs.

Cette liste de différences entre RCA et GCA révèle que ces deux approches ont quelques similitudes notamment : l'utilisation du quantificateur $\exists$, le traitement des relations unaires et binaires, ainsi que le calcul des concepts unaires. Il est très important de noter que RCA et GCA ne modélisent pas les relations de la même façon. Bien que le sens des relations soit indiqué dans le contexte graphe (graphe orienté) le processus de GCA traite automatiquement les relations dans les deux sens (direct et inverse), tandis que RCA prend en compte les relations telles qu'elles sont explicitement définies dans la Famille Relationnelle de Contextes (FRC), c'est-à-dire des domaines vers les codomaines.

La section suivante illustre, à travers un exemple, la différence de modélisation des relations par les deux approches RCA et GCA.

7.2 Modélisation des relations dans RCA et GCA

Bien que les relations soient orientées dans les contextes graphes GCA les considère dans les deux sens. En revanche, dans RCA, pour qu'une relation soit prise en compte à la fois dans le sens direct et dans le sens inverse, il est nécessaire de l'indiquer explicitement dans la FRC en ajoutant la relation directe ainsi que son inverse. Pour élucider ce point, nous reprenons l'exemple FRC_1 ($\mathbf{K} = \{\mathcal{K}_{Person}, \mathcal{K}_{Car}, \mathcal{K}_{Garage}\}$, $\mathbf{R} = \{owner, sell, maintain\}$) présenté dans le tableau 3.1 (chapitre 3), à propos des personnes, des voitures et des garages. La démarche consiste à présenter, puis à confronter les résultats obtenus avec RCA et GCA sur cet exemple.

La figure 7.1 illustre le contexte graphe CG_1 [1] correspondant à FRC_1. Pour une visualisation qui met en avant la définition d'un contexte graphe en tant qu'une relation d'incidence entre des tuples d'objets et des attributs, le tableau 7.1 met en évidence la représentation tabulaire [2] des relations binaires de CG_1. FRC_1 (tableau 3.1) et CG_1 (figure 7.1) sont deux structures qui représentent les mêmes informations : les garages qui vendent et/ou entretiennent les voitures appartenant à des personnes. La définition des relations $sell(O_{Garage}, O_{Car})$, $maintain(O_{Garage}, O_{Car})$ et $owner(O_{Car}, O_{Person})$ pour RCA précise l'orientation des relations et signifie que les concepts de $\mathcal{K}_{Garage}$ et de $\mathcal{K}_{Car}$ contiendront des attributs relationnels dans leurs intensions en plus des attributs unaires. En revanche, les concepts de $\mathcal{K}_{Person}$ n'auront que des attributs unaires dans leurs intensions, car les objets O_{Person} de $\mathcal{K}_{Person}$ ne constituent le domaine (la source) d'aucun contexte relationnel.

Tableau 7.1 – Représentation tabulaire des relations binaires de CG_1.

(*Garage, Car*)	*maintain*	*sell*
(A, car1)	×	×
(A, car6)	×	×
(D, car2)	×	
(D, car5)	×	
(C, car2)		×
(C, car5)		×
(B, car3)	×	×
(B, car4)	×	×

(*Car, Person*)	*owner*
(car1, Bob)	×
(car6, Bob)	×
(car2, Julie)	×
(car5, Julie)	×
(car3, Charlie)	×
(car4, Alice)	×

En termes de résultats, référons-nous au tableau 3.10 qui récapitule le nombre de concepts par contexte objets-attributs au fil des itérations de RCA sur FRC_1. Il en ressort que le ré-

1. CG est mis pour Contexte Graphe

2. Ces deux tableaux, fusionnés en power Contexte family [Wille, 1997], correspondraient à K^2 : le contexte des relations binaires.

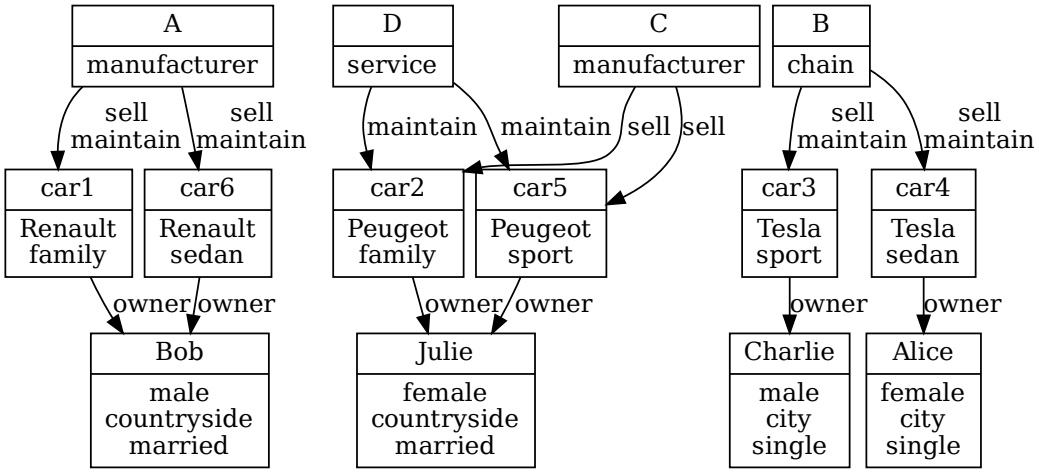

FIGURE 7.1 – Contexte graphe CG_1 correspondant à FRC_1.

sultat RCA sur FRC_1, que nous notons par RCA(FRC_1) compte un total de 40 concepts : 10 concepts pour $\mathcal{K}_{Person}$, 17 concepts pour $\mathcal{K}_{Car}$ et 13 concepts pour $\mathcal{K}_{Garage}$. Dans cette étude comparative, similairement à l'outil de GCA qui n'affiche pas les bottom ($\bot$) concepts lorsqu'elles ont une extension vide, nous ne les considérons pas dans le nombre de concepts (pour RCA et GCA), ce qui ramène à 37 concepts pour RCA(FRC_1). Le résultat d'exécution de GCA sur CG_1, noté GCA(CG_1), compte 7 patterns constitués d'un total de 45 [3] concepts unaires. Cet ensemble de patterns est présenté à la figure 7.2.

Comme nous l'avons déjà souligné, GCA traite systématiquement les relations à la fois dans leur sens direct et dans leur sens inverse. Pour illustrer, considérons la hiérarchie des concepts sur les personnes, présentée à la figure 7.3. On remarque que les concepts des personnes contiennent des liens vers les concepts des voitures. À titre d'exemple, l'intension du concept $Q3d$ (figure 7.3) contient les arêtes [*owner a* _] et [*owner b* _]. L'arête [*owner a* _] représente la relation *owner* entre les nœuds a ($Q3a$) et d ($Q3d$) ; et l'arête [*owner b* _] représente la relation *owner* entre les nœuds b ($Q3b$) et d du pattern Q3 (figure 7.2). Cela signifie que les instances du concept $Q3d$ sont en réalité caractérisées par la relation $owner^{-1}$ avec les nœuds a et b du pattern Q3. En effet, les instances du concept $Q3d$ sont propriétaires des voitures des concepts $Q3a$ et $Q3b$. Plus précisément, Bob est propriétaire des voitures $car6$ ($Q3a$) et $car1$ ($Q3b$). Notons que ces arêtes doivent être lues comme faisant partie d'un même pattern. Par exemple, les arêtes [*owner a* _] dans $Q3d$ et [*owner a* _] dans $Q5c$ sont distinctes. L'une renvoie à $Q3a$ et l'autre renvoie à $Q5a$. Leur généralisation dans leur

3. Les top ($\top$) concepts qui ne sont pas affichés par l'outil de GCA (celui des *Garages* dans ce cas) sont rajoutés dans le nombre de concepts

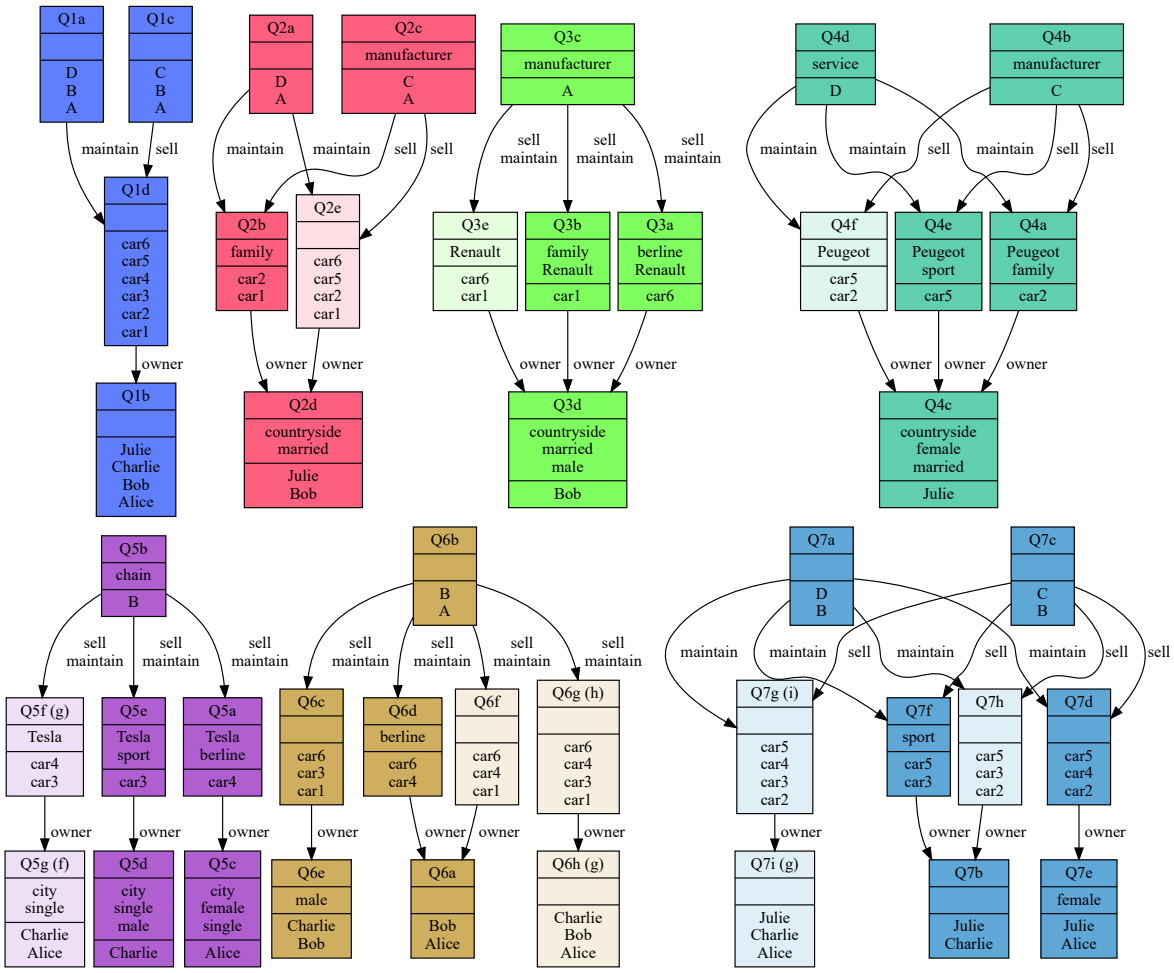

Figure 7.2 – Ensemble des graph patterns du contexte graphe CG_1.

sur-concept $Q6a$ apparaît sous la forme de l'arête [*owner d* _] renvoyant au concept $Q6d$.

La hiérarchie de concepts des personnes provenant de GCA(CG_1) (figure 7.3) compte 13 concepts contre 9 concepts pour le treillis des personnes (figure 7.4) issu de RCA(FRC_1). Il convient de préciser que les 9 concepts de RCA(FRC_1) possèdent chacun un équivalent dans GCA(CG_1), du point de vue des extensions. Les 4 concepts supplémentaires $Q6a, Q6h, Q7b$ et $Q7i$ qui n'ont pas d'équivalents dans RCA(FRC_1) sont strictement décrits par leur relation $owner^{-1}$ vers les concepts voitures dont ils sont respectivement propriétaires (voir les patterns Q6 et Q7 - figure 7.2). D'où la création de 4 concepts supplémentaires dans la hiérarchie des voitures (décrits inversement par leur relation *owner* vers les concepts $Q6a, Q6h, Q7b$ et $Q7i$ respectivement) par rapport au treillis des voitures de RCA(FRC_1). Finalement, en comparant les extensions des concepts, les 37 concepts de RCA(FRC_1) sont inclus dans les 45 concepts de GCA(CG_1) : **RCA(FRC_1)** $\subset$ **GCA(CG_1)**. Les 8 concepts sup-

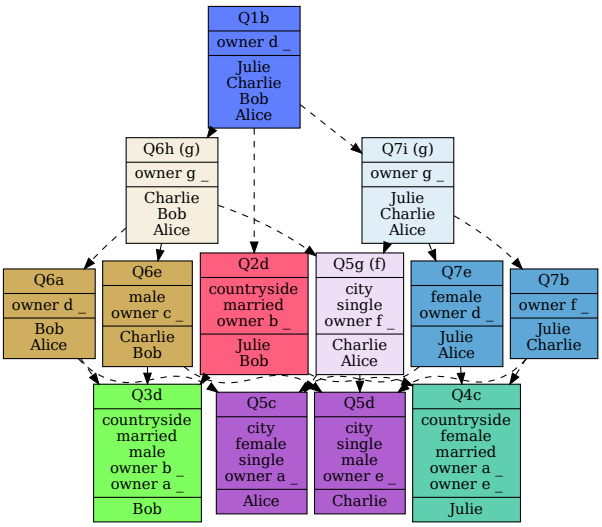

FIGURE 7.3 – Hiérarchie des concepts *Person* issue de GCA(CG_1).

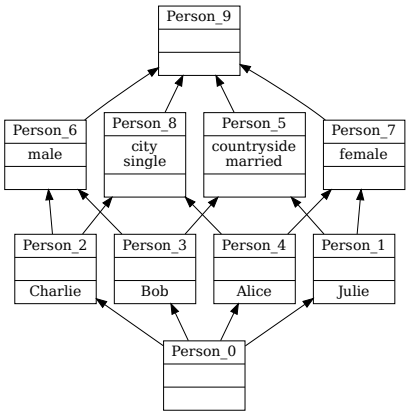

FIGURE 7.4 – Treillis de $\mathcal{K}_{Person}$ issu de RCA(FRC_1).

plémentaires de GCA(CG_1) sont liés à l'intégration automatique des relations inverses dans le processus de GCA. Donc, intuitivement, l'ajout des relations inverses dans les données de RCA devrait permettre de rapprocher les résultats de RCA de ceux de GCA.

Dans la suite, afin d'examiner plus en profondeur la différence entre les deux approches, nous les comparerons à partir de leurs **points communs**, à savoir :

- le calcul des concepts unaires, car RCA se limite aux concepts unaires,
- l'usage du quantificateur $\exists$ pour RCA, car GCA ne dispose que du quantificateur $\exists$ implicitement,
- le traitement des données contenant uniquement des relations unaires et binaires, car les relations $n(>2)$-aires nécessitent des modélisations supplémentaires dans RCA [KEIP et al., 2019],
- l'ajout des relations inverses dans les données de RCA, car GCA intègre automatiquement les relations inverses dans son processus.

La section suivante présente une comparaison empirique de RCA et GCA dans ce cadre commun, sur différents exemples.

7.3 Comparaison empirique de RCA et GCA

Nous avons précédemment constaté que certains concepts de GCA, non produits par RCA, sont liés à l'intégration automatique des relations inverses dans le processus de GCA.

On s'attend à ce que l'ajout des relations inverses dans les données de RCA rapproche autant que possible ses résultats de ceux de GCA. La question qui se pose est alors de déterminer dans quelle mesure ce rapprochement est effectif. Nous comparons dans un premier temps RCA(FRC) et RCA(FRC_r), où FRC_r = FRC + relations inverses, avec FRC une Famille Relationnelle de Contextes. Ensuite, nous présentons d'une part deux exemples pour lesquels RCA(FRC_r) ≡ GCA(GC) et d'autre part, nous présentons trois exemples pour lesquels RCA(FRC_r) ≢ GCA(GC), GC étant le contexte graphe correspondant à la FRC considérée. Cette comparaison empirique fait partie des travaux présentés dans [Fokou et al., 2024a].

7.3.1 Ajout des relations inverses à une FRC

L'ajout de relations inverses à une FRC entraîne un ajout d'informations dans les données, donc l'ajout de nouveaux attributs relationnels qui peut conduire à la modification des intensions des concepts existants ou à la construction de nouveaux concepts. Ceci est cohérent avec le processus de RCA, car il y a toujours au moins autant de concepts dans un treillis à l'itération $i + 1$ qu'à l'itération i. Formellement, pour toute FRC, on a :

$$RCA(FRC) \subseteq RCA(FRC\_r) \tag{7.1}$$

Pour l'exemple FRC_1, l'ajout des relations inverses se fait en ajoutant 3 contextes relationnels : *sell_r*(O_{Car}, O_{Garage}), *owner_r*(O_{Person}, O_{Car}) et *maintain_r*(O_{Car}, O_{Garage}) qui sont respectivement l'inverse des relations *sell, owner* et *maintain*. La FRC résultante est désignée par FRC_1_r, avec la chaîne "_r" (pour *reverse*) qui matérialise les inverses tant dans les noms des relations que dans les noms des FRCs. Pour illustration, la relation *owner_r*(O_{Person}, O_{Car}) définit le fait qu'*une personne possède une voiture*. Ensuite, dans le processus de RCA, les concepts construits sur le contexte $\mathcal{K}_{Car}$, par association avec l'opérateur de scaling et la relation *owner_r*, seront utilisés pour former des attributs relationnels afin d'étendre le contexte $\mathcal{K}_{Person}$. Dans la suite, une FRC_i étendue par l'ajout des relations inverses est désignée par FRC_i_r.

Le résultat RCA(FRC_1) compte 37 concepts qui sont tous inclus dans RCA(FRC_1_r) qui compte 45 concepts. Pour exemple, les figures 7.4 et 7.5 présentent les treillis de $\mathcal{K}_{Person}$ construits respectivement à partir de FRC_1 et FRC_1_r. Il en résulte 4 concepts supplémentaires {**Person_10**, **Person_11**, **Person_12**, **Person_13**} dans le deuxième treillis (figure 7.5), en raison des attributs relationnels de la forme $\exists$ *owner_r*(Car_i) qui étendent le contexte $\mathcal{K}_{Person}$. Ces concepts supplémentaires sont strictement dûs à l'ajout des relations inverses, car comme nous pouvons le constater, ce sont tous des concepts introduits par les attributs relationnels de la forme $\exists$ *owner_r*(Car_i). Dans cette même logique, ces 4 concepts supplémentaires vont générer 4 attributs relationnels qui vont à leur tour induire 4 nouveaux concepts des voitures {**Car_17**, **Car_18**, **Car_19**, **Car_20**}. Ces nouveaux concepts des voitures sont strictement décrits de manière inverse via la relation *owner* pointant sur les concepts {**Person_10**, **Person_11**, **Person_12**, **Person_13**} comme mis en évi-

dence dans la figure 7.6, qui montre le treillis $\mathcal{K}_{Car}$ issu de RCA(FRC$_1$_r) sans le bottom concept [4]. Par ailleurs, remarquons la présence des attributs relationnels induits par les relations inverses *sell_r* et *maintain_r* qui enrichissent les intensions des concepts de $\mathcal{K}_{Car}$. Pour ce qui est du contexte $\mathcal{K}_{Garage}$, son nombre de concepts est resté inchangé, mais les intensions de certains concepts ont été modifiées pour intégrer les attributs relationnels induits par les nouveaux concepts de $\mathcal{K}_{Car}$.

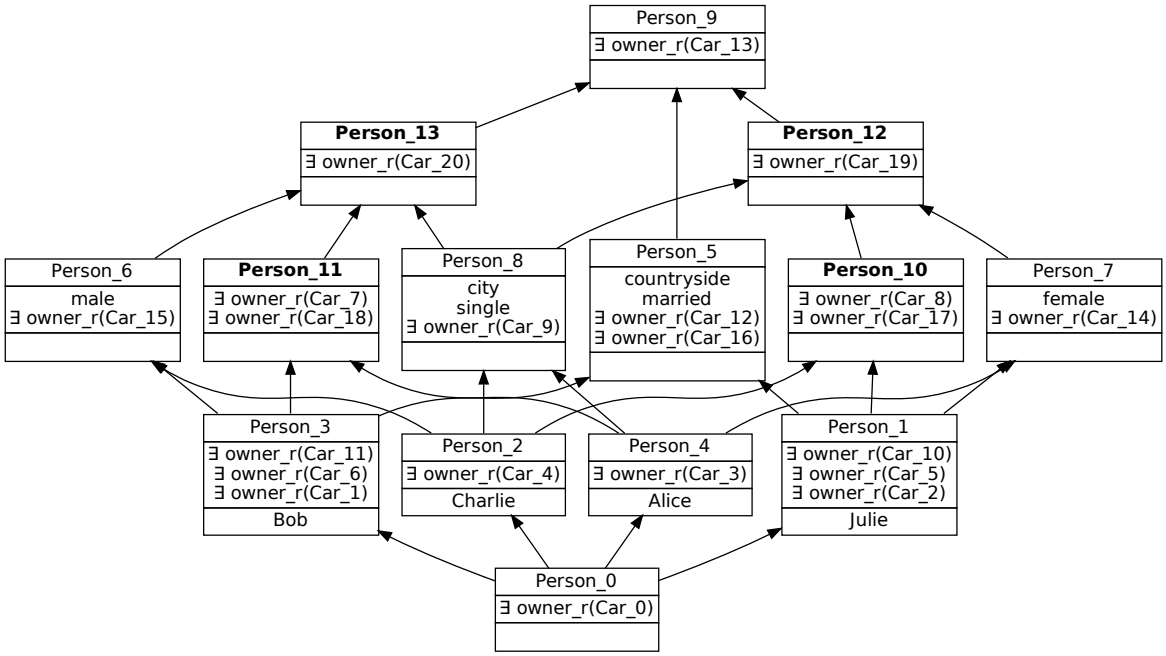

FIGURE 7.5 – Treillis de $\mathcal{K}_{Person}$ issu de RCA(FRC$_1$_r).

7.3.2 Équivalence entre les résultats de GCA et de RCA avec relation inverses

Une comparaison des résultats RCA(FRC$_1$_r) et GCA(CG$_1$) montre une équivalence du point de vue des extensions de concepts. À titre d'illustration, examinons les concepts de $\mathcal{K}_{Person}$ représentés dans les treillis de la figure 7.5 pour RCA et de la figure 7.3 pour GCA. Les deux treillis ont le même nombre de concepts et les concepts ont les mêmes extensions. Il en est de même pour les treillis de voitures et des garages. En résumé, pour cet exemple, on montre que, lorsque l'on applique RCA à FRC$_1$_r contenant les relations inverses, le résultat RCA(FRC$_1$_r) est équivalent au résultat obtenu avec GCA sur le contexte graphe

4. Le concept bottom ($\perp$) a été supprimé. Il en sera de même pour le reste des treillis de ce chapitre, pour des raisons de lisibilité.

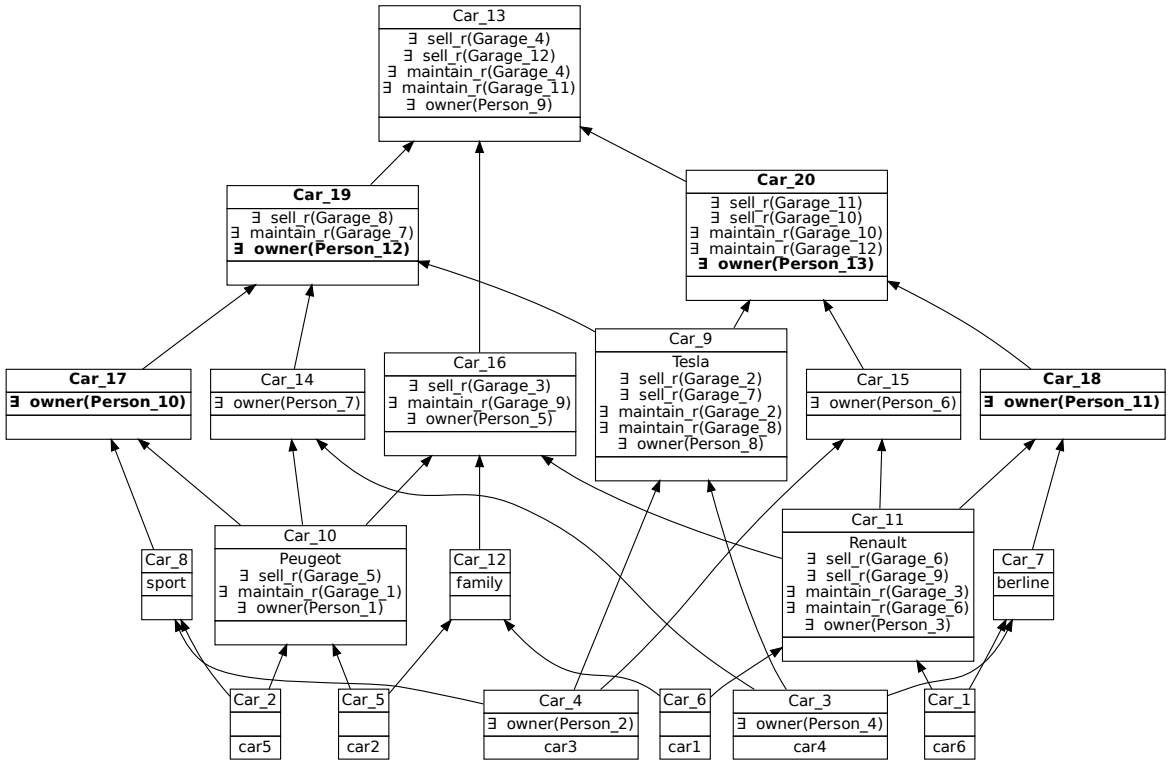

Figure 7.6 – Treillis de $\mathcal{K}_{Car}$ issu de RCA(FRC_1_r).

CG_1 correspondant :

$$RCA(FRC_1\_r) \equiv GCA(CG_1) \tag{7.2}$$

Considérons à présent un second exemple, à savoir celui de la famille royale britannique présenté dans le chapitre 4 (figure 4.1) et désignons-le par CG_2. Le tableau 7.2 illustre la FRC_2 correspondant à CG_2. La FRC_2 est constituée d'un contexte objets-attributs *Personne* qui décrit les personnes par leur genre et un contexte objets-objets (*has-parent*) qui indique qu'une personne a pour parent une autre personne. Cet exemple a la particularité d'être constitué d'une seule relation dont le domaine et le codomaine sont identiques, mais il est néanmoins nécessaire d'y ajouter la relation inverse. L'analyse de ce second exemple s'inscrit dans la même logique que celle de l'exemple précédent.

La figure 7.7 montre le treillis de concepts RCA(FRC_2) qui compte 8 concepts, regroupant les personnes suivant leur genre et suivant le fait qu'ils aient des parents. La hiérarchie de concepts issue de GCA(CG_2) a déjà été présentée à la figure 4.13, mais pour faciliter la comparaison, la figure 7.8 reprend cette hiérarchie de concepts. En comparaison avec RCA(FRC_2), GCA(CG_2) compte 18 concepts (le $\top$ concept y compris) contre 8 concepts pour RCA(FRC_2) : **RCA($\mathbf{FRC_2}$)** $\subseteq$ **GCA($\mathbf{CG_2}$)**. En effet, GCA(CG_2) contient aussi le regroupement des personnes selon qu'ils aient des enfants, ce qui est rendu possible grâce à l'intégration

TABLEAU 7.2 – FRC_2 (**K** = {*Personne*}, **R** = {*has-parent*}) correspondant au contexte graphe CG_2.

Personne	male	female
Georges	×	
Charlotte		×
William	×	
Harry	×	
Kate		×
Diana		×
Charles	×	

has-parent	*Georges*	*Charlotte*	*William*	*Harry*	*Kate*	*Diana*	*Charles*
Georges			×		×		
Charlotte			×		×		
William						×	×
Harry						×	×
Kate							
Diana							
Charles							

automatique des relations inverses. À titre d'illustration, les concepts $Q4i$, $Q4g$ et $Q3d$ sont des exemples de concepts qui regroupent des personnes ayant des enfants.

Comme dans l'exemple précédent, on ajoute les relations inverses pour se mettre au même niveau d'information que dans GCA. La FRC_2_r est construite en ajoutant la relation *has-child* comme inverse de la relation *has-parent*. Cette relation inverse a tout son sens dans cet exemple, car si x a pour parent y alors y a naturellement pour enfant x. Le résultat RCA(FRC_2_r) est présenté à la figure 7.9 et compte 18 concepts. Comme nous pouvons le constater, ce treillis contient bien entendu aussi des regroupements des personnes (concepts) suivant le fait qu'ils ont des enfants. Par exemple, les concepts **family_9**, **family_16** et **family_11** mis en évidence en gras (figure 7.9), correspondent respectivement aux concepts $Q4i$, $Q4g$ et $Q3d$ de GCA(CG_2). Notons que le $\top$ concept $family\_3$ de RCA(FRC_2_r) a une intension vide, c'est ce qui justifie pourquoi il n'est pas affiché du côté de GCA (figure 7.8). En tout, avec l'ajout de la relation inverse *has-child*, RCA et GCA produisent les mêmes concepts pour cet exemple :

$$RCA(FRC_2) \subseteq RCA(FRC_2\_r) \equiv GCA(CG_2) \tag{7.3}$$

Ces deux exemples suggèrent que l'ajout de relations inverses dans les données de RCA conduit à des résultats équivalents pour les deux approches. Toutefois, comme nous le verrons dans la section suivante, il existe des cas où, malgré l'ajout de relations inverses dans les données de RCA, les deux approches donnent des résultats distincts : certains concepts obtenus en GCA ne sont pas générés par RCA.

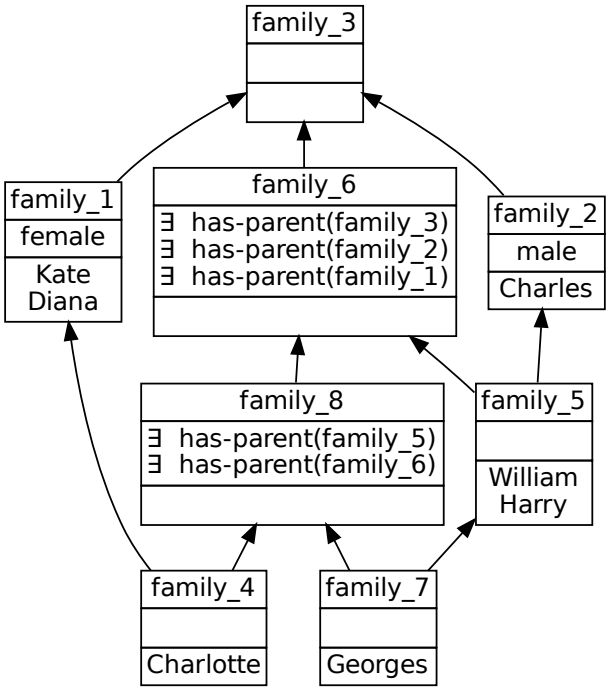

Figure 7.7 – Treillis de concepts RCA(FRC$_2$).

7.3.3 Non-équivalence entre les résultats de RCA et GCA

Les exemples que nous traitons ici sont des variantes du contexte graphe CG$_1$ (figure 7.1) obtenus en effectuant de petites modifications sur les connexions entre les objets pour avoir différents schémas de données. Nous présentons ci-dessous les 3 variantes considérées et désignées respectivement par CG$_{1a}$, CG$_{1b}$ et CG$_{1c}$. Les FRCs associées, intégrant les relations inverses sont respectivement désignées par FRC$_{1a}$_r, FRC$_{1b}$_r et FRC$_{1c}$_r.

RCA(FRC$_{1a}$_r) $\not\equiv$ GCA(CG$_{1a}$). Le contexte graphe CG$_{1a}$ est obtenu en modifiant la relation *maintain* dans CG$_1$ (figure 7.1) pour que la voiture $car6$ soit entretenue par le $garage\ D$ au lieu du $garage\ A$, comme mis en évidence dans la figure 7.10. CG$_{1a}$ est alors composé de deux sous-graphes comparé à CG$_1$ qui se compose de trois graphes. En termes d'informations, on peut dire que CG$_1$ contient les garages (A et B) qui *vendent et entretiennent* leurs voitures, les garages (C) qui *vendent uniquement* des voitures et les garages (D) qui *entretiennent uniquement* les voitures - le $garage\ D$ fait l'entretien des voitures vendus par des garages (C) *qui vendent uniquement* les voitures. Suite à la modification effectuée sur la relation *maintain* pour obtenir CG$_{1a}$, on a désormais les garages qui *entretiennent les voitures vendus par des garages qui vendent et entretiennent les voitures* - le $garage\ D$ entretient la voiture $car6$ vendue par le *garage* A qui vend et entretient la voiture $car1$.

Concernant les résultats, GCA(CG$_{1a}$) compte 53 concepts contre 52 pour RCA(FRC$_{1a}$_r),

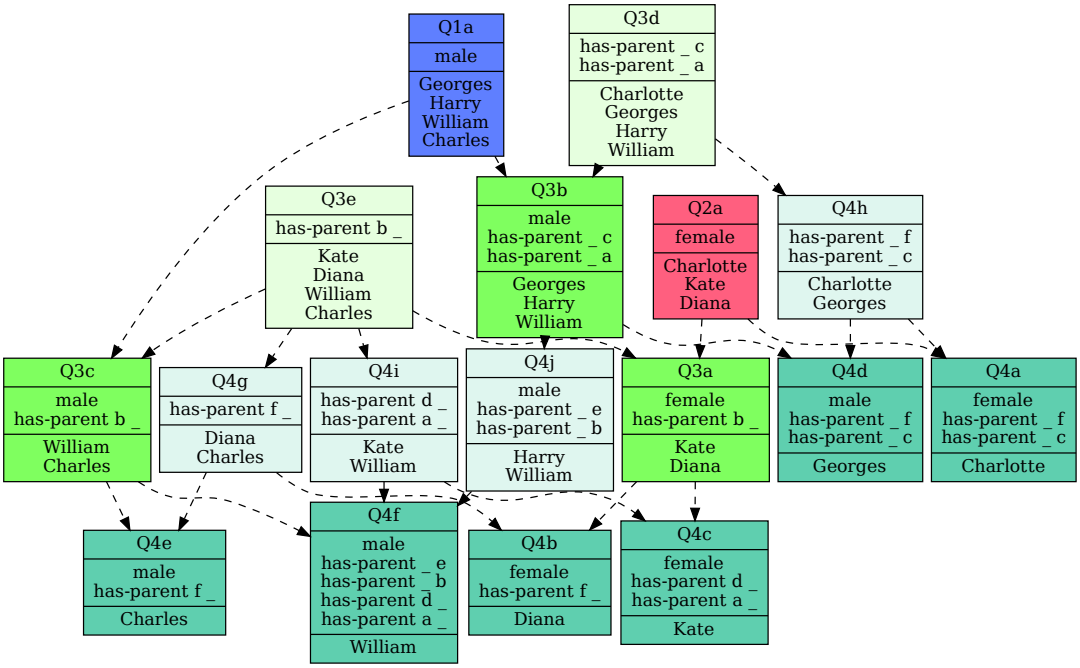

FIGURE 7.8 – Hiérarchie de concepts de CG_2.

ce qui fait un concept **supplémentaire** du côté de GCA par rapport à RCA. La figure 7.11 montre le PGP $Q2z = ((z), P2z)$ décrivant le concept d'extension $\{car5, car2, car1\}$ (en vert) qui identifie ce concept supplémentaire. Le pattern $P2z$ désigne le sous-graphe incluant le nœud z, les nœuds principaux (*core nodes*) - en rouge - ainsi que l'ensemble des autres nœuds nécessaires à la description de z, mentionnés entre parenthèses dans l'identifiant (nom) du concept $Q2z$ *(s m k l o bc)*. Il s'agit de la notation parenthésée décrite dans le chapitre 4 consacré à GCA.

Nous décrivons maintenant ce qui caractérise ce concept supplémentaire. Comme le montre la figure 7.11, les nœuds adjacents à z sont les nœuds bc, l et o introduits par les relations : *sell(bc, z)*, *maintain(l, z)* et *owner(z, o)*. Les autres nœuds sont introduits par ces nœuds adjacents (les adjacents des adjacents de z) et ainsi de suite jusqu'aux nœuds principaux (*core nodes*) qui portent l'information principale du pattern. Une description simplifiée de z peut être formulée comme suit : z est le concept des voitures vendues par un garage bc, entretenues par un garage l et appartenant à une personne o, qui possède également une voiture familiale (*family*) s et une autre voiture m, toutes deux vendues par le même garage bc. Cette description inclut le cycle *(bc, s, o, m, bc)*. À ce stade, nous soupçonnons que ce cycle pourrait être la raison pour laquelle RCA ne construit pas ce concept.

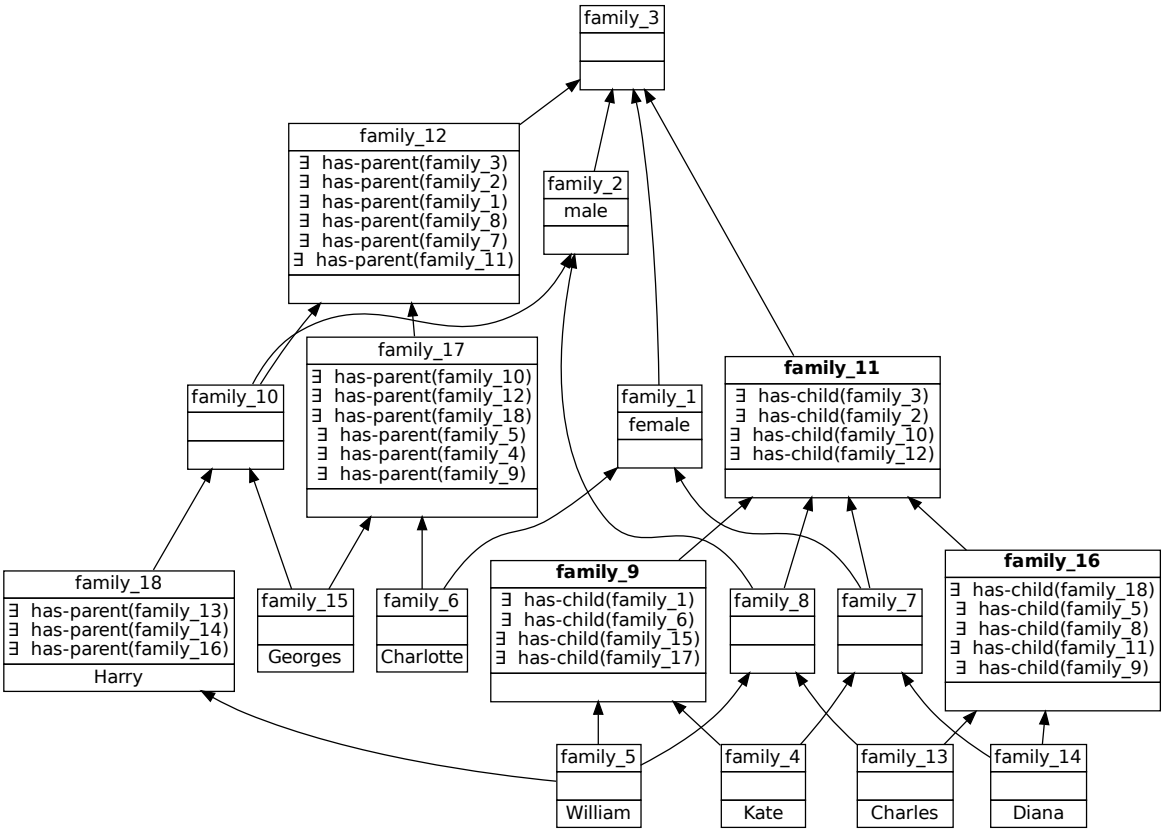

Figure 7.9 – Treillis de concepts RCA(FRC_2_r).

Enfin, en termes de complexité et de facilité d'interprétation, GCA(CG_1) ne contient pas de concepts automorphes, contrairement à GCA(CG_{1a}) qui en possède 17, c'est-à-dire 17 doublons du point de vue des extensions de concepts. Pour ce qui est du nombre de graph patterns, CG_1 produit 7 patterns (figure 7.2) contre 3 graph patterns pour CG_{1a}. Concrètement, les graph patterns de GCA(CG_1) sont plus petits (à l'image des motifs présents dans CG_1) et donc plus simples à comprendre par rapport à ceux de GCA(CG_{1a}).

RCA(FRC_{1b}_r) $\not\equiv$ GCA(CG_{1b}). La deuxième variante CG_{1b} (figure 7.12) est obtenue de CG_1 suite à la même modification effectuée sur la relation *maintain* pour obtenir CG_{1a}, en plus d'une modification de la relation *owner*. Dans le contexte graphe CG_{1b} tel que le montre la figure 7.12, la voiture $car2$ qui avait pour propriétaire $Julie$ appartient dorénavant à $Charlie$ et inversement, $car3$ qui appartenait à $Charlie$ devient une propriété de $Julie$. La remarque principale en ce qui concerne la structure du contexte graphe, est qu'il est composé d'un seul graphe plutôt que de plusieurs (petits) graphes comme dans CG_1 ou CG_{1a}. Au niveau des résultats, GCA(CG_{1b}) contient 66 concepts contre 64 concepts pour RCA(FRC_{1b}_r), soit **deux concepts en plus du côté de GCA**, car les 64 concepts de RCA ont leurs concepts équivalents dans les concepts de GCA. Les 66 concepts de GCA(CG_{1b})

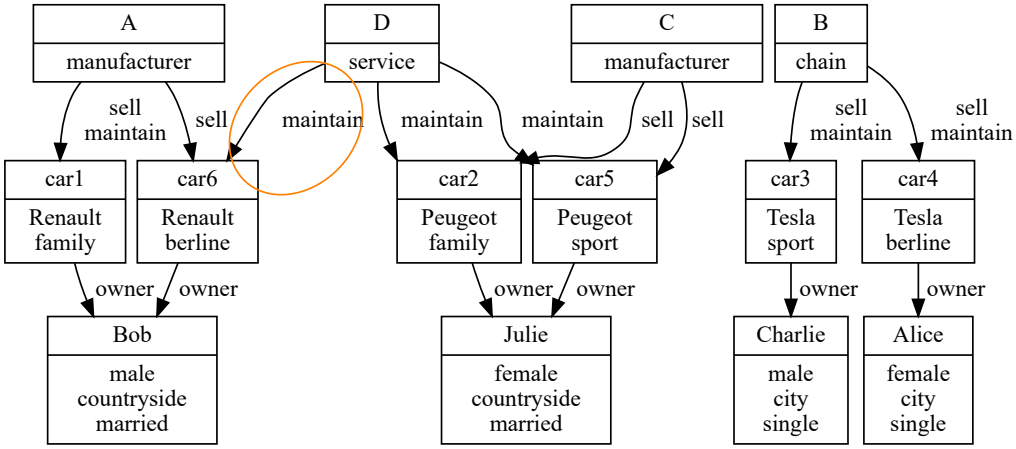

FIGURE 7.10 – Contexte graphe CG_{1a}.

sont regroupés en **un seul graph pattern**, tandis que CG_1 produit 7 patterns et CG_{1a} en produit 3. Par ailleurs, GCA(CG_{1b}) contient 230 concepts automorphes, soit plus du triple du nombre de concepts unaires, ce qui augmente singulièrement la taille du pattern (qui a au total 66 + 230 nœuds) et par conséquent sa difficulté d'interprétation. D'un point de vue pratique, même si les concepts automorphes sont mis de côté, il est difficile d'interpréter en langage naturel ce que pourrait exprimer un graph pattern constitué de 66 concepts.

RCA(FRC$_{1c}$_r) $\not\equiv$ GCA(CG$_{1c}$). La troisième variante CG_{1c} est obtenue de CG_1 par suppression du $garage\ D$ et par ajout d'une nouvelle personne ($Jean$). Les relations (*owner, sell* et *maintain*) ont été réorganisées comme illustré à la figure 7.13. Ce nouveau contexte graphe est composé de deux graphes comme CG_{1a}. GCA(CG_{1c}) contient 50 concepts contre 48 concepts pour RCA(FRC$_{1c}$_r), soit **deux concepts en plus du côté de GCA**. GCA(CG_{1c}) contient 27 concepts automorphes et l'ensemble de ses concepts est regroupé en 3 graph patterns (au même titre que CG_{1a}) contre 7 graph patterns pour CG_1.

Une analyse des concepts de GCA(CG_{1b}) et de GCA(CG_{1c}) qui ne sont pas produits du côté de RCA a été faite et le point commun entre ces concepts de GCA est la *présence des cycles dans leurs descriptions*, ce qui peut expliquer le fait que RCA ne construit pas ces concepts.

À partir de ces observations, on constate que plus le contexte graphe est connecté (c'est-à-dire avec un nombre réduit de composantes), comme dans CG_{1b} (figure 7.12), plus GCA est susceptible de produire des patterns complexes[5], et plus le nombre de concepts automorphes tend à augmenter. Ceci s'explique par le fait que GCA dans son fonctionnement,

5. En termes de nombre de nœuds et de connexions entre ces nœuds.

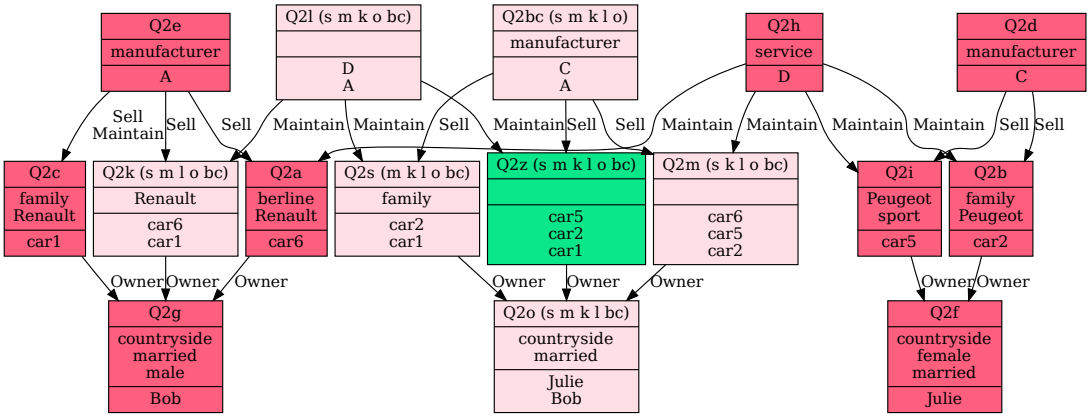

Figure 7.11 – PGP $Q2z$ avec z comme tuple de projection.

capture les structures du contexte graphe ainsi que les structures combinées qui en découlent. A contrario, pour un contexte graphe composé d'un ensemble de graphes simples [6] comme dans CG_1 (figure 7.1), GCA produit des patterns relativement simples, à l'image des motifs présents dans le contexte graphe.

Ajout des relations inverses dans GCA. L'impact de l'ajout des relations inverses dans les données de GCA a été examiné sur les cinq exemples CG_1, CG_2, CG_{1a}, CG_{1b} et CG_{1c}. Comme pour RCA, les relations inverses (arêtes binaires inverses) ont été intégrées dans chacun de ces contextes graphes, afin d'analyser d'éventuelles différences dans les résultats de GCA par rapport à ceux obtenus à partir des données initiales (sans arêtes inverses). Tout naturellement, les relations inverses sont perçues par GCA comme des doublons en ce sens qu'elles n'apportent pas de nouvelles informations, car logiquement, GCA intègre déjà automatiquement ces informations inverses au cours de son processus. Cette observation peut être formulée par : **GCA(CG) ≡ GCA(CG_r)**. Pour illustration, la figure 7.14 montre le graph pattern Q1 issu de GCA(CG_1) et de GCA(CG_1_r) respectivement. Nous pouvons remarquer que l'unique différence entre ces deux patterns se trouve au niveau des connexions entre les concepts.

En résumé, l'ensemble de ces observations et analyses permettent de conjecturer que l'ensemble des extensions de concepts de RCA est inclus dans l'ensemble des extensions de concepts de GCA. Aussi, l'ajout des relations inverses dans les données de RCA permet de réduire l'écart entre les résultats des deux approches. Le tableau 7.3 établit une synthèse de l'ensemble des résultats de ces expériences en récapitulant pour chaque exemple le nombre de concepts, le nombre de graph patterns, ainsi que le nombre de concepts automorphes (doublons). Le bloc en vert met en évidence les exemples pour lesquels il y a équivalence de

6. En termes de taille et de structure des connexions.

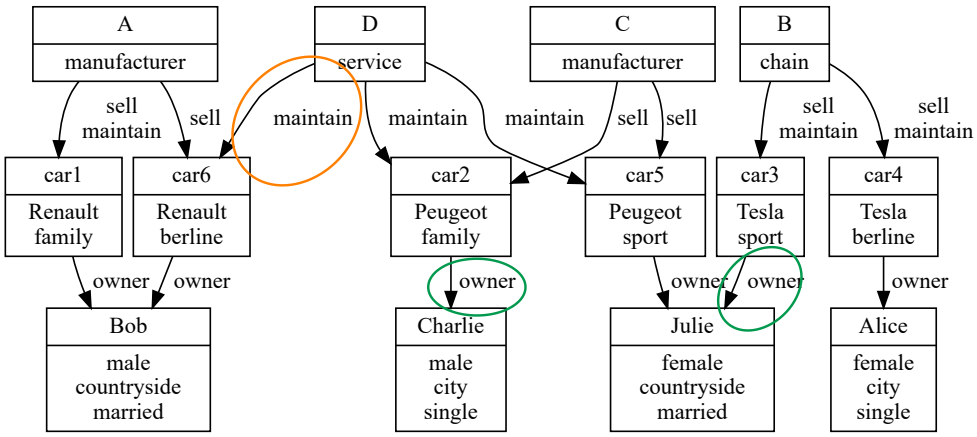

FIGURE 7.12 – Contexte graphe CG_{1b}.

résultats entre RCA (avec intégration des relations inverses) et GCA. Inversement, le bloc en orange délimite les exemples pour lesquels GCA produit des concepts en plus par rapport à RCA, c'est-à-dire le résultat de RCA est strictement inclus dans celui de GCA malgré l'ajout des relations inverses. En comparant les nombres de concepts des colonnes **RCA** et **GCA**, l'impact des relations inverses par rapport aux nombres de concepts de **RCA_r** est clairement perceptible.

TABLEAU 7.3 – Récapitulatif du nombre de concepts, de graph patterns et de concepts automorphes (doublons).

Exemple	**RCA**	**RCA_r**	**GCA**	**Patterns**	**Doublons**
CG_1	37	45	45	7	0
CG_2	8	18	18	4	0
CG_{1a}	40	52	53	3	17
CG_{1b}	41	64	66	1	230
CG_{1c}	33	48	50	3	27

La section suivante présente les démonstrations formelles qui viennent corroborer les observations issues des exemples précédents.

7.4 Comparaison théorique

Nous poursuivons notre analyse en présentant dans cette section, une étude théorique relative à la comparaison entre RCA et GCA au regard de leurs extensions de concepts.

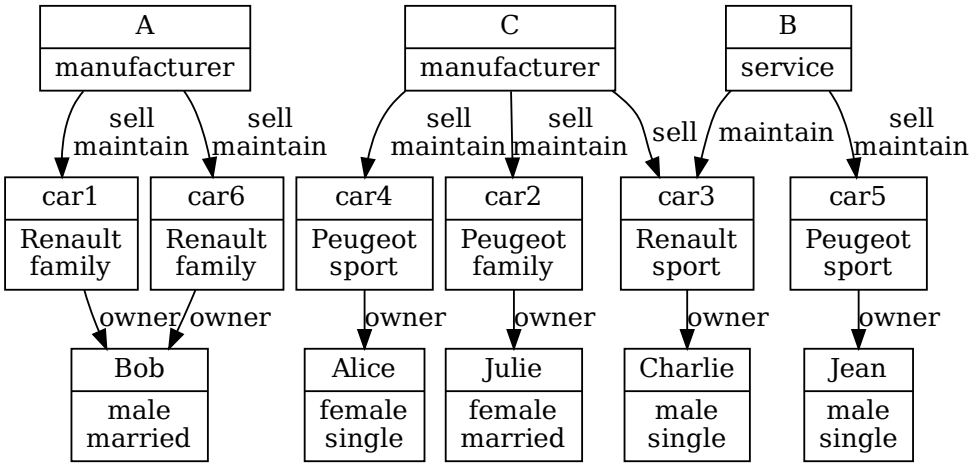

Figure 7.13 – Contexte graphe CG_{1c}.

Comme nous ne considérons que les concepts unaires, cela revient à comparer des ensembles d'ensembles d'objets. En effet, les ensembles d'objets sont les mêmes des deux côtés, bien qu'ils soient divisés en contextes formels dans le modèle initial de données en entrée de RCA. Étant donné C_R l'ensemble des extensions de concepts RCA, et C_G l'ensemble des extensions de concepts GCA, nous démontrons que $C_R \subseteq C_G$, et que l'inverse n'est pas vrai. Avant d'énoncer et de démontrer ce résultat, nous établissons d'une part, quelques observations concernant les concepts RCA par rapport à leur processus de génération itératif et d'autre part la traduction d'une FRC en un contexte graphe équivalent afin de guider la comparaison. Cette comparaison théorique fait partie des travaux présentés dans [Fokou et al., 2025b].

7.4.1 Notion de rang et d'intension initiale d'un concept RCA

Le processus de génération de concepts RCA est itératif. À chaque itération, de nouveaux concepts peuvent être définis et les intensions des concepts existants peuvent être enrichies, grâce à de nouveaux attributs relationnels dérivés des nouveaux concepts de l'itération précédente. Un concept étant identifié par son extension, on peut dire qu'un nouveau concept est une nouvelle extension. Le processus itératif s'arrête lorsqu'aucun nouveau concept n'est créé, ce qui se produit après un nombre fini d'étapes, car l'ensemble des objets est fini, et donc l'ensemble des extensions possibles est également fini.

RCA considère toujours les concepts après la dernière itération, c'est-à-dire les paires (X, Y) où Y est l'intension finale. Nous introduisons deux nouvelles notions : le *rang* d'un

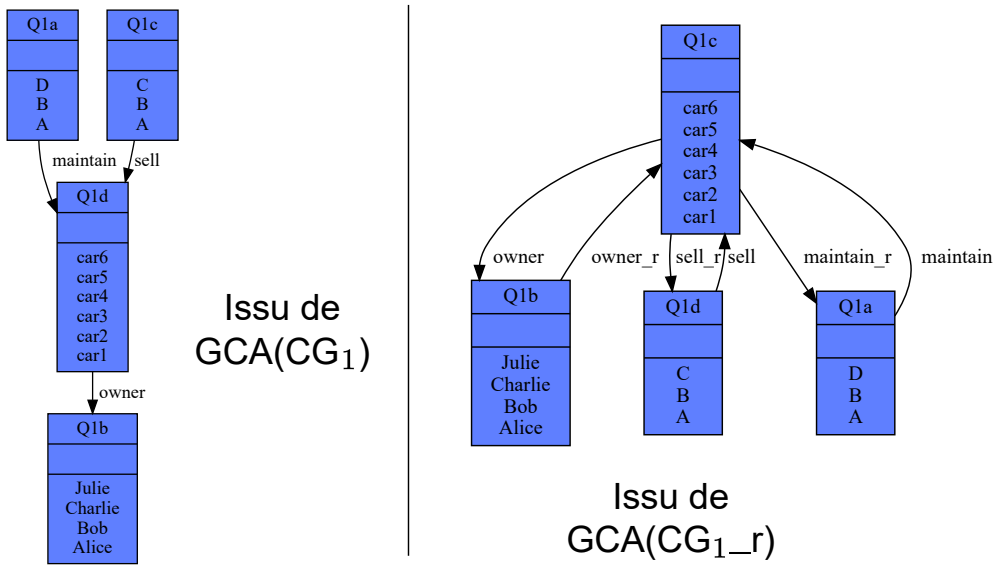

FIGURE 7.14 – Graph pattern Q1 issu de GCA(CG_1) et son équivalent issu de GCA(CG_1_r).

concept et l'*intension initiale* d'un concept.

Définition 7.1 (Rang d'un concept). Le *rang* d'un concept $C = (X, Y)$ de RCA, noté $rank(C)$, est l'itération RCA $i \geq 0$ qui génère pour la première fois un concept avec l'extension X.

Définition 7.2 (Intension initiale). L'*intension initiale* d'un concept $C = (X, Y)$ de RCA, notée $\widehat{Int}(C)$ ou simplement $\widehat{Y}$, est l'intension de C à l'itération $rank(C)$, c'est-à-dire lorsque l'extension X a été générée pour la première fois. Nous avons donc $\widehat{Y}' = X$.

Par définition des itérations RCA qui ne peuvent qu'ajouter de nouveaux attributs relationnels, nous avons $\widehat{Y} \subseteq Y$. Nous pouvons également dire que l'intension initiale d'un concept C ne peut se référer qu'à des concepts C_i (via des attributs relationnels) tels que $rank(C_i) < rank(C)$, car un concept ne peut être utilisé avant d'avoir été créé. Ces résultats préliminaires sont importants car les intensions initiales brisent les cycles qui peuvent exister dans les intensions finales des concepts. Cela établit donc un ordre topologique entre les concepts via leur rang, tout en préservant la caractérisation des extensions puisque $X = \widehat{Y}'$.

7.4.2 Transposition des FRCs en contextes graphes

Afin d'avoir une base de comparaison solide, une Famille Relationnelle de Contextes (FRC) du côté de RCA doit d'abord être traduite en un contexte graphe équivalent du côté de GCA. Cela est relativement simple lorsque l'on se limite à leur base commune, car les

deux approches disposent des données multi-relationnelles. La principale subtilité concerne l'ensemble des objets qui est partitionné dans RCA (via les contextes objets-attributs) et non dans GCA. Cela conduit à l'introduction d'attributs unaires supplémentaires pour représenter les différentes catégories d'objets.

Définition 7.3. Soit $(\mathbf{K}, \mathbf{R})$ une FRC. Le contexte graphe $K = (O, A, I)$ correspondant est défini comme suit :

- $O := \bigcup_{i=1..n} O_i$: l'ensemble des objets est l'union disjointe des ensembles d'objets de tous les contextes objets-attributs $\mathcal{K}_i \in \mathbf{K}$;
- $A = A_1 \cup A_2$: l'ensemble des attributs est composé d'attributs unaires (A_1) et d'attributs binaires (A_2), où :
  - $A_1 := \bigcup_{i=1..n}(\{a_{\mathcal{K}_i}\} \cup A_i)$: l'ensemble des attributs unaires est l'union disjointe des ensembles d'attributs de tous les contextes objets-attributs, étendu avec le nom des contextes ($a_{\mathcal{K}_i}$) comme types (catégories) d'objets ;
  - $A_2 := \{r_k\}_{k=1..m}$: l'ensemble des attributs binaires est l'ensemble des noms de relations, un pour chaque contexte objets-objets ;
- La relation d'incidence entre les tuples d'objets et les attributs est définie comme suit :

$$\begin{aligned} I \;:=\; & \{a_{\mathcal{K}_i}(o) \mid \mathcal{K}_i \in \mathbf{K}, o \in O_i\} \\ \cup\; & \{a(o) \mid o \in O_i, a \in A_i, (o, a) \in I_i\} \\ \cup\; & \{r(o_1, o_2) \mid r \in \mathbf{R}, (o_1, o_2) \in r\} \end{aligned}$$

Par exemple, le contexte graphe correspondant à FRC_2 à propos de la famille royale, présentée dans le tableau 7.2 est égale à $K = (O, A, I)$ avec :

- $O = O_{\textit{Personne}}$
- $A = A_1 \cup A_2$ où $A_1 = \{a_{\textit{Personne}}\} \cup A_{\textit{Personne}}$ et $A_2 = \{\textit{has-parent}\}$
- $\begin{aligned} I \;=\; & \{a_{\textit{Personne}}(o) \mid o \in O_{\textit{Personne}}\} \\ \cup\; & \{a(o) \mid (o, a) \in I_{\textit{Personne}}\} \\ \cup\; & \{\textit{has-parent}(o_1, o_2) \mid (o_1, o_2) \in \textit{has-parent}\} \end{aligned}$

Notons qu'aucune information n'est perdue au cours du processus de traduction , et la traduction peut être facilement inversée tant que les attributs $a_{\mathcal{K}_i}$ sont différenciés des autres attributs unaires. Cependant, il n'existe pas de traduction générale évidente des contextes graphes vers les FRCs en raison de l'absence de catégories d'objets et de l'existence possible de relations n-aires.

7.4.3 Comparaison des ensembles d'extensions de concepts

Nous démontrons ci-après que toutes les extensions de concepts RCA sont également des extensions de concepts GCA. Cette hypothèse a été émise précédemment à la section 7.3

à partir des observations faites sur les exemples, où il a également été observé que l'inverse n'est pas toujours vrai. Afin d'établir cette preuve, nous construisons d'abord un *Projected Graph Pattern* (PGP) qui simule l'intension initiale d'un concept RCA, puis nous démontrons deux lemmes concernant cette construction.

Définition 7.4 (PGP simulant un concept RCA). Soit C un concept dans une famille de treillis de concepts RCA. Soit $\mathcal{C} = \{C_k\}$ le sous-ensemble des concepts qui participent à la description de C, c'est-à-dire les concepts référencés dans les attributs relationnels de l'intension initiale de C, et de manière récursive dans l'intension initiale de ces concepts. $\mathcal{C}$ inclut également C lui-même en tant que C_0. Nous définissons le PGP $Q = (x_0, P_0)$ simulant C en introduisant un ensemble de variables $\mathcal{X} = \{x_k\}$, contenant une variable distincte pour chaque concept $C_k \in \mathcal{C}$; et en définissant un pattern P_k pour chaque concept C_k tel que :

$$\begin{aligned} P_k \;:=\; & \{a_{\mathcal{K}_i}(x_k) \mid C_k \text{ appartient au treillis de concepts du contexte } \mathcal{K}_i\} \\ \cup\; & \{a(x_k) \mid a \in \widehat{Int}(C_k)\} \\ \cup\; & \bigcup \{\{r(x_k, x_l)\} \cup P_l \mid \exists r(C_l) \in \widehat{Int}(C_k)\} \end{aligned}$$

Nous définissons $Q_k = (x_k, P_k)$ comme le PGP qui simule C_k, et nous notons cette relation de simulation $Q_k \sim C_k$.

La définition récursive de P_k est bien définie grâce à l'ordre topologique basé sur le rang des intensions initiales. Le cas de base est constitué de concepts de rang 0, qui n'ont aucun attribut relationnel dans leur intension initiale.

Pour illustration, définissons le PGP qui simule le concept $family\_8$ du treillis de la figure 7.7. Pour des raisons de lisibilité, les concepts de ce treillis sont renommés C_{fi} pour chaque concept $family\_i$, par exemple C_{f8} pour le concept $family\_8$. Supposons[7] que $\widehat{Int}(C_{f8}) = \{\exists \textit{has-parent}(C_{f5}), \exists \textit{has-parent}(C_{f6})\}$ et trouvons $Q_{f8} = (x_8, P_{f8})$, le PGP qui simule C_{f8}. L'intension initiale de chacun des concepts C_{f5} et C_{f6} contient les attributs relationnels référençant les concepts $\{C_{f3}, C_{f2}, C_{f1}\}$. Par conséquent, l'ensemble des concepts qui participent à la description de C_{f8} est $\mathcal{C} = \{C_{f5}, C_{f6}, C_{f3}, C_{f2}, C_{f1}\}$. On a :

$$\begin{aligned} P_{f8} \;=\; & \{a_{\textit{Personne}}(x_8)\} \\ \cup\; & \{\textit{has-parent}(x_8, x_5)\} \cup P_{f5} \\ \cup\; & \{\textit{has-parent}(x_8, x_6)\} \cup P_{f6} \end{aligned}$$

Avec :

$$\begin{aligned} P_{f5} \;=\; & \{a_{\textit{Personne}}(x_5), \textit{male}(x_5)\} \\ \cup\; & \{\textit{has-parent}(x_5, x_3)\} \cup P_{f3} \\ \cup\; & \{\textit{has-parent}(x_5, x_2)\} \cup P_{f2} \\ \cup\; & \{\textit{has-parent}(x_5, x_1)\} \cup P_{f1} \end{aligned} \qquad \begin{aligned} P_{f6} \;=\; & \{a_{\textit{Personne}}(x_6)\} \\ \cup\; & \{\textit{has-parent}(x_6, x_3)\} \cup P_{f3} \\ \cup\; & \{\textit{has-parent}(x_6, x_2)\} \cup P_{f2} \\ \cup\; & \{\textit{has-parent}(x_6, x_1)\} \cup P_{f1} \end{aligned}$$

7. Factuellement, l'intension de C_{f8} contient aussi l'intension de C_{f6}. De plus l'intension initiale de C_{f8} est égale à son intension finale.

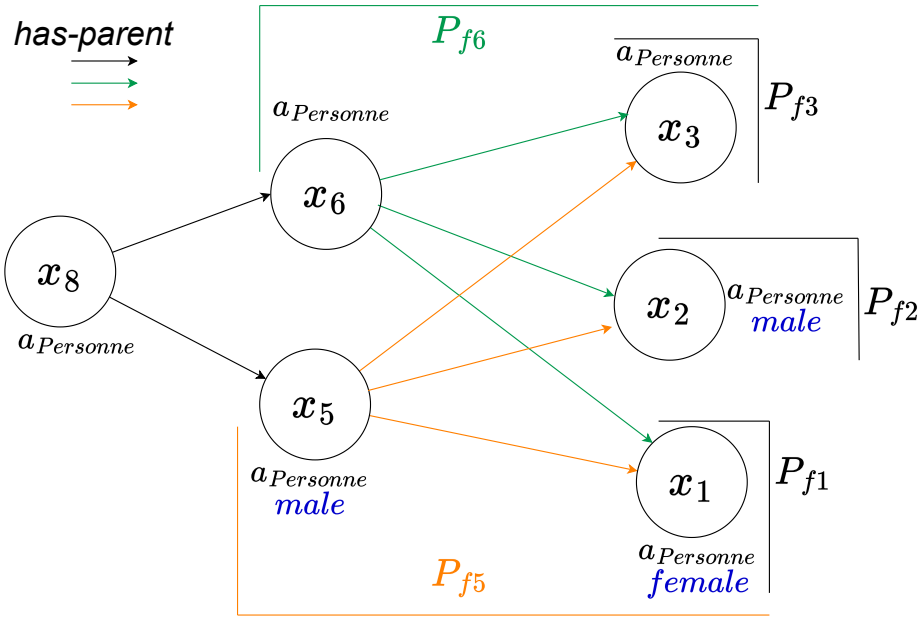

FIGURE 7.15 – Représentation graphique du pattern P_{f8} qui simule le concept C_{f8}.

Où : $P_{f1} = \{a_{Personne}(x_1), \textit{female}(x_1)\}$, $P_{f2} = \{a_{Personne}(x_2), \textit{male}(x_2)\}$ et $P_{f3} = \{a_{Personne}(x_3)\}$.

La figure 7.15 illustre la représentation graphique du pattern P_{f8} dans lequel les flèches reliant les nœuds représentent la relation *has-parent*. P_{f8} fournit en réalité la description du concept C_{f8} identifié par la variable x_8, c'est-à-dire : les personnes (x_8) qui ont pour parents des personnes (x_5, x_6) ayant eux aussi des parents (x_1, x_2, x_3). Donc x_8 est la variable qui identifie (représente) les personnes ayant un grand-parent, c'est le cas de $Charlotte$ et $Georges$, ce qui justifie le résultat $Ext(C_{f8}) = \{Charlotte, Georges\}$.

Lemme 1. Sous les conditions de la définition 7.4, nous avons pour tout $C_k \in \mathcal{C}$

$$Q_k \sim C_k \Longrightarrow ext(Q_k) = Ext(C_k).$$

Démonstration. Nous prouvons ce lemme par récurrence sur le rang des concepts dans $\mathcal{C}$. Dans le cas général, nous considérons un concept C_k de rang n, appartenant au contexte $\mathcal{K}_i$. Nous supposons que le lemme est vrai pour tous les concepts de rang inférieur à n, en particulier ceux qui apparaissent dans l'intension initiale $\widehat{Y}_k$ de C_k.

(1) Nous prouvons d'abord que $Ext(C_k) \subseteq ext(Q_k)$. En considérant un certain $o_k \in Ext(C_k)$, nous devons prouver que $o_k \in ext(Q_k)$, qui est défini par $ext(Q_k) = \{o \mid Q_k \subseteq_q (o, I)\}$. Ainsi, $o_k \in ext(Q_k)$ est vrai si et seulement $Q_k \subseteq_q (o_k, I)$, c'est-à-dire qu'il existe une application ϕ des variables aux objets telle que $\phi(x_k) = o_k$ et $\phi(P_k) \subseteq I$ (voir définition 4.7 sur

l'inclusion des PGPs). Nous avons $o_k \in Ext(C_k) = \widehat{Y}_k' = \bigcap_{\alpha \in \widehat{Y}_k} \alpha'$ [8], c'est-à-dire que $o_k \in \alpha'$ pour tous les éléments α de l'intension initiale. Il existe deux types d'éléments :

- Attributs formels ($\alpha = a \in A_i$) : $o_k \in \alpha'$ implique que $o_k \in a'$, donc $(o_k, a) \in I_i$.
- Attributs relationnels ($\alpha = \exists r(C_l)$) : $o_k \in \alpha'$ implique ici que $o_k \in (\exists r(C_l))'$, ce qui implique qu'il existe un objet o_l tel que $(o_k, o_l) \in r$ et $o_l \in Ext(C_l)$. En appliquant l'hypothèse de récurrence, nous obtenons $o_l \in ext(Q_l)$, et donc il existe une application ϕ_l telle que $\phi_l(x_l) = o_l$ et $\phi_l(P_l) \subseteq I$.

Nous définissons l'application $\phi := \{x_k \to o_k\} \cup \bigcup_l \phi_l$ comme l'union des applications de chaque attribut relationnel plus l'application de x_k à o_k. Il reste donc à montrer que $\phi(P_k) \subseteq I$. P_k peut être décomposé en sous-ensembles d'éléments suivants :

- $a_{\mathcal{K}_i}(x_k)$: o_k appartient au contexte $\mathcal{K}_i$, donc nous avons $a_{\mathcal{K}_i}(o_k) \in I$ selon la définition du contexte graphe qui correspond à la FRC.
- $a(x_k)$: par définition de Q_k, nous avons $a \in \widehat{Y}_k$, donc comme vu ci-dessus, nous avons $(o_k, a) \in I_i$, donc $a(o_k) \in I$.
- $r(x_k, x_l)$: par définition de Q_k, nous avons $\exists r(C_l) \in \widehat{Y}_k$, donc comme vu ci-dessus, nous avons $(o_k, o_l) \in r$, donc $r(o_k, o_l) \in I$.
- $P_l \subseteq P_k$: nous avons vu plus haut que $\phi_l(P_l) \subseteq I$, donc par définition de ϕ qui inclut ϕ_l, nous avons également $\phi(P_l) \subseteq I$.

Comme toutes les parties de P_k se projettent sur des parties de I via ϕ, on obtient $\phi(P_k) \subseteq I$.

(2) Nous prouvons ensuite que $ext(Q_k) \subseteq Ext(C_k)$. En considérant $o_k \in ext(Q_k)$, nous savons qu'il existe une application ϕ des variables aux objets telle que $\phi(x_k) = o_k$ et $\phi(P_k) \subseteq I$. Nous devons prouver que $o_k \in Ext(C_k) = \widehat{Y}_k' = \bigcap_{\alpha \in \widehat{Y}_k} \alpha'$. Nous devons donc prouver que o_k appartient à l'extension de chaque élément α de l'intension initiale. Nous le faisons pour les deux types d'éléments :

- $\alpha = a \in A_i$ (attribut formel) : par définition de Q_k, nous savons que $a(x_k) \in P_k$. À partir de $\phi(P_k) \subseteq I$, nous obtenons $\phi(a(x_k)) \in I$, puis à partir de $\phi(x_k) = o_k$, nous obtenons $a(o_k) \in I$. Nous obtenons donc également $(o_k, a) \in I_i$ par traduction du contexte, et par conséquent, $o_k \in a'$.
- $\alpha = \exists r(C_l)$ (attribut relationnel) : par définition de Q_k, nous savons que $r(x_k, x_l) \in P_k$ et $P_l \subseteq P_k$. À partir des contraintes sur ϕ, nous pouvons déduire que $r(o_k, o_l) \in I$, avec $o_l = \phi(x_l)$, et $\phi(P_l) \subseteq I$. À partir de là, nous pouvons conclure que $o_l \in ext(Q_l)$, en projetant simplement ϕ sur les variables de P_l. En appliquant l'hypothèse de récurrence, nous obtenons $o_l \in Ext(C_l)$. À partir de $r(o_k, o_l) \in I$, nous obtenons également $(o_k, o_l) \in r$ par traduction du contexte. Nous avons donc $o_k \in (\exists r(C_l))'$, par définition de l'extension d'un attribut relationnel.

8. L'opérateur $'$ correspond à l'opérateur de dérivation qui à un ensemble d'attributs associe l'ensemble d'objets qui partagent ces attributs.

Le cas de base est pour $rank(C_k) = 0$. La même preuve que dans le cas général peut être utilisée, car l'hypothèse de récurrence n'est utilisée que pour les attributs relationnels, et les concepts de rang 0 n'en ont aucun. □

Théorème 2. Soient $(\mathbf{K}, \mathbf{R})$ une FRC et K son contexte graphe correspondant. Pour chaque concept RCA sur $(\mathbf{K}, \mathbf{R})$, il existe un concept GCA sur K ayant la même extension.

Démonstration. Considérons un concept C dans une famille de treillis de concepts et le PGP Q qui simule C, c'est-à-dire tel que $Q \sim C$. Comme cas particulier du lemme précédent avec $k = 0$, nous avons $ext(Q) = Ext(C)$. Dans GCA, le couple (ext, int) est une connexion de Galois, donc $(ext(Q), int(ext(Q)))$ forme un concept, et donc $ext(Q)$ est l'extension d'un concept GCA. □

7.4.4 Comparaison en cas d'ajout des relations inverses dans les données de RCA

Les observations faites sur les exemples de la section 7.3 ont montré que l'ajout des relations inverses aux données de RCA permet de rapprocher les résultats RCA de ceux de GCA, car GCA prend automatiquement en compte les relations inverses dans son processus. Formellement, pour chaque relation $r \in \mathbf{R}$ dans la FRC $(\mathbf{K}, \mathbf{R})$, une nouvelle relation $r^- = \{(o_j, o_i) \mid (o_i, o_j) \in r\}$ est incluse dans l'ensemble des relations. Nous notons $\mathbf{R}^-$ l'ensemble des relations inverses dérivées d'un ensemble de relations $\mathbf{R}$, sachant que l'inverse d'une relation est nommé en rajoutant la chaîne "_r" à la fin du nom de la relation (comme en section 7.3.1).

La question que nous abordons dans cette section est de savoir si le résultat précédent indiquant l'inclusion des extensions RCA dans les extensions GCA reste vrai avec l'ajout des relations inverses dans les données de RCA. Nous démontrons que l'inclusion reste valide en montrant que l'ensemble des extensions GCA n'est pas affecté par l'introduction des arêtes binaires inverses dans les données de GCA.

Lemme 2. Soient $K = (O, A, I)$ un contexte de graphe et $K^- = (O, A^-, I^-)$ son extension avec les relations inverses, c'est-à-dire $A^- = A \cup \mathbf{R}^-$, où $\mathbf{R} \subseteq A$ est l'ensemble des attributs binaires dans A, et $I^- = I \cup \{r^-(o_j, o_i) \mid r(o_i, o_j) \in I\}$. Il existe une correspondance biunivoque entre les concepts de K et les concepts de K^- :

$$(R, Q) \in GCA(K) \iff (R, Q^-) \in GCA(K^-),$$

où, étant donné $Q = (\overline{x}, P)$, nous avons $Q^- = (\overline{x}, P^-)$ et $P^- = P \cup \{r^-(y, x) \mid r(x, y) \in P\}$.

Démonstration. Le calcul des intensions des concepts GCA se résume à des produits catégoriques répétés de la relation d'incidence. En particulier, lorsque l'arête $r(x, y)$ appartient à $I \times \ldots \times I$ (n fois), cela signifie que pour tout $i \in 1..n$, il existe une paire d'objets (u_i, v_i) telle que $r(u_i, v_i) \in I$. Par définition de I^-, il s'ensuit que pour tout $i \in 1..n$, nous avons également $r^-(v_i, u_i) \in I^-$. À partir de là, nous concluons que le produit catégorique $I^- \times \ldots \times I^-$ contient $r^-(y, x)$ en plus de $r(x, y)$. En effet, le produit catégorique définit $x = (u_1, \ldots, u_n)$ et $y = (v_1, \ldots, v_n)$. L'inverse est également vrai. Chaque fois que $r^-(y, x) \in I^- \times \ldots \times I^-$, nous avons $r(x, y) \in I \times \ldots \times I$ car chaque fois que $r^-(v_i, u_i) \in I^-$ est vrai, $r(u_i, v_i) \in I \subseteq I^-$.

Cela implique que les concepts de K^- peuvent être dérivés des concepts de K, simplement en ajoutant des arêtes binaires inverses dans les patterns selon la définition ci-dessus de P^-. Les extensions de concepts sont les mêmes. □

Sur la base de ce lemme, le Théorème 2 précédent peut être étendu à une FRC intégrant les relations inverses.

Théorème 3. Soient $(\mathbf{K}, \mathbf{R})$ une FRC et K son contexte graphe correspondant. Pour chaque concept RCA de la FRC étendue $(\mathbf{K}, \mathbf{R} \cup \mathbf{R}^-)$, il existe un concept GCA de K avec la même extension.

Démonstration. Il est facile de voir que si K est le contexte graphe correspondant à la FRC $(\mathbf{K}, \mathbf{R})$, alors K^- est le contexte graphe correspondant à la FRC étendue avec les relations inverses. Par conséquent, d'après le théorème 2, toutes les extensions des concepts de la FRC étendue sont des extensions des concepts de K^-. Le Lemme 2 montre que K et K^- ont les mêmes extensions de concepts. Par conséquent, toutes les extensions des concepts de la FRC étendue sont des extensions de concepts du contexte graphe non étendu K. □

En résumé, les démonstrations confirment que l'ensemble des extensions des concepts produit par RCA est inclus dans celui des concepts de GCA. De plus, l'intégration explicite des relations inverses dans GCA s'avère superflue, puisqu'elle n'induit aucun changement dans les résultats.

7.5 Conclusion

Dans ce chapitre, nous avons mené une étude comparative de RCA et de GCA du point de vue des extensions de concepts. Cette analyse a mis en évidence que, si les deux approches présentent de nombreuses différences, elles partagent néanmoins certains points communs, notamment l'utilisation du quantificateur $\exists$, le calcul des concepts unaires, ainsi que le traitement des relations binaires. Par ailleurs, la prise en compte automatique des relations inverses dans le processus de GCA est compensée du côté de RCA par l'ajout manuel

de ces relations dans les données. Cette intégration des relations inverses en RCA permet ainsi de rapprocher ses résultats de ceux de GCA.

Sur la base de ces points communs, ce chapitre a d'abord montré, à travers plusieurs exemples, que l'ensemble des extensions de concepts de RCA est inclus dans celui de GCA. Autrement dit, pour tout concept RCA, il existe un concept GCA possédant la même extension. Ce constat empirique a ensuite été étayé par une démonstration théorique établissant formellement que l'ensemble des extensions des concepts RCA est effectivement inclus dans l'ensemble des extensions des concepts GCA. Par ailleurs, il a également été démontré que l'ajout des relations inverses dans les données de GCA n'entraîne aucun changement sur l'ensemble des concepts, hormis l'introduction d'arêtes redondantes.

Le chapitre suivant propose une comparaison intensionnelle de RCA et GCA, complétant ainsi la deuxième dimension nécessaire à la mise en parallèle de l'ensemble des concepts RCA avec celui de GCA.

Chapitre 8

Comparaison intensionnelle de RCA et GCA dans leur cadre commun

Sommaire

Ce chapitre est consacré à la comparaison des intensions des concepts RCA et GCA dans leur cadre commun. Nous y démontrons que les intensions des concepts RCA sont incluses dans celles des concepts GCA. Nous commençons par préciser la notion d'intension pour un concept RCA et pour un concept GCA dans la section 8.1. Ensuite, la section 8.2 présente le processus de transformation d'une famille de treillis RCA en un ensemble de patterns relationnels comparables aux graph patterns GCA. Le rapprochement entre les patterns relationnels RCA et les graph patterns GCA est ensuite effectué dans la section 8.3. La section 8.4 aborde la comparaison intensionnelle proprement dite et démontre que l'ensemble des patterns relationnels issus de RCA est inclus dans l'ensemble des graph patterns GCA. Enfin, la section 8.5 met en évidence l'apport pratique des patterns relationnels RCA, au-delà de leur rôle de passerelle pour cette étude comparative. Cette étude comparative de RCA et GCA, sous l'angle de leurs intensions de concepts, s'inscrit dans le cadre des travaux publiés dans [Fokou et al., 2025b].

8.1 Intensions des concepts RCA et GCA

Afin de bien expliciter la notion d'intension pour un concept RCA et un concept GCA, cette section détaille successivement l'intension d'un concept RCA, puis celle du concept GCA correspondant, c'est-à-dire le concept GCA possédant la même extension.

Pour ce faire, nous utilisons l'exemple de la FRC du tableau 8.1, désignée par FRC_3 qui porte sur les animaux et les aliments qu'ils consomment. Le contexte graphe CG_3 correspondant est présenté à la figure 8.1. La figure 8.2 présente la famille de treillis obtenue sur FRC_3 et la figure 8.3 illustre l'ensemble des graph patterns GCA calculé sur CG_3. Notre illustration de la notion d'intension de concept porte sur le concept *animal* d'extension $\{a1, a4\}$, dont le nom (identifiant) est $animals\_2$ du côte de RCA (figure 8.2) et $Q3c$ du côté de GCA (figure 8.3).

Tableau 8.1 – FRC_3 (**K** = $\{\mathcal{K}_{Animal}, \mathcal{K}_{Food}\}$, **R** = $\{eat\}$) à propos des animaux et de leurs aliments.

$\mathcal{K}_{Animal}$	herbivore	carnivore
a1	×	
a2		×
a3		×
a4	×	

$\mathcal{K}_{Food}$	fruit	grass	meat
f1	×		
f2		×	
f3		×	
f4	×		
f5			×
f6			×

eat	f1	f2	f3	f4	f5	f6
a1	×	×	×	×		
a2					×	×
a3					×	×
a4	×	×	×	×		

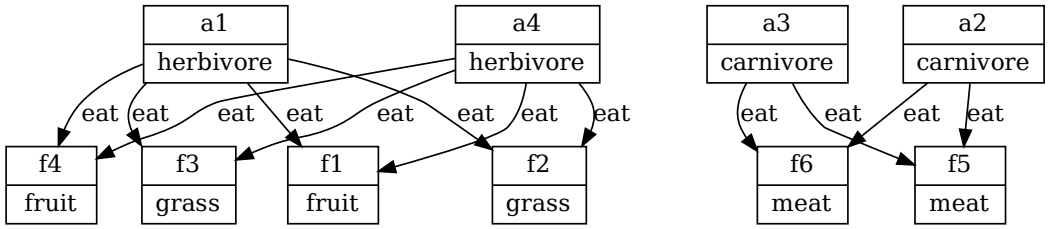

Figure 8.1 – Contexte graphe CG_3 correspondant à FRC_3 (figure 8.1).

8.1.1 Intension d'un concept RCA

L'intension d'un concept RCA se compose, d'une part, des attributs unaires caractérisant les objets du concept, et d'autre part, des attributs relationnels qui capturent les re-

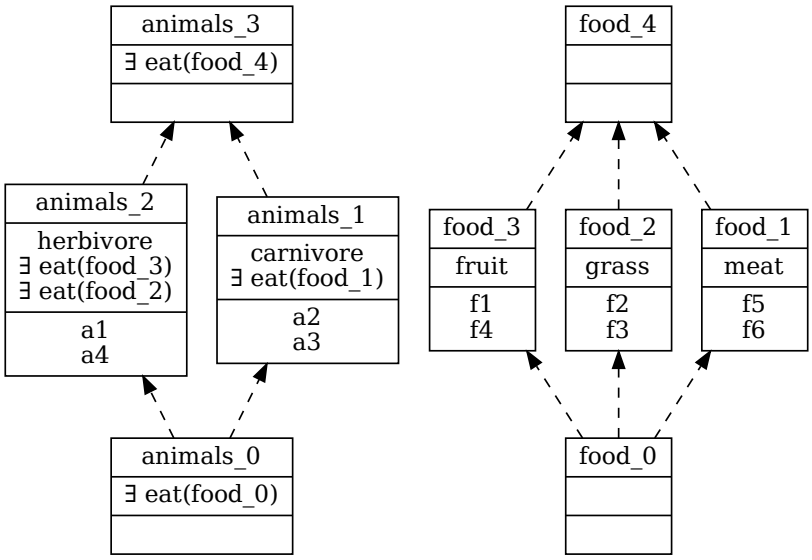

FIGURE 8.2 – Famille de treillis RCA obtenue sur FRC_3.

lations entre ces objets et d'autres objets. Par conséquent, l'interprétation complète d'une intension de concept requiert de naviguer dans la famille de treillis au moyen des attributs relationnels qui relient les concepts entre eux. Les chemins de navigation sont déterminés par les attributs relationnels de l'intension du concept à analyser et, de manière récursive, par l'intension des concepts référencés par ces attributs relationnels.

Pour illustration, l'intension complète du concept $animals\_2$ (figure 8.2) est l'ensemble $\{herbivore, \exists eat(food\_2), \exists eat(food\_3), \exists eat(food\_4)\}$ qui contient trois attributs relationnels pointant sur les concepts du contexte $\mathcal{K}_{Food}$. L'ensemble des concepts qui participent à l'interprétation de l'intension du concept $animals\_2$ est illustré à la figure 8.4 sous forme d'un graphe orienté qui met en évidence les connexions entre ces concepts. Ce graphe représente les différents chemins de navigation au sein de la famille de treillis, conduisant à l'interprétation de l'intension du concept $animals\_2$. Par exemple, le chemin menant au concept $food\_3$ indique que les animaux de $animals\_2$ (les herbivores) se nourrissent de fruits. Notons qu'il s'agit ici d'un cas simple, puisque le treillis de $\mathcal{K}_{Food}$ ne contient pas d'attributs relationnels. En revanche, dans l'exemple FRC_1 (tableau 3.1) relatif aux garages, voitures et personnes, les intensions des concepts de $\mathcal{K}_{Garage}$ font référence à des concepts de $\mathcal{K}_{Car}$, dont les intensions renvoient à leur tour à des concepts de $\mathcal{K}_{Person}$, comme l'illustre l'extrait de la figure 3.11.

Un tel "graphe navigation" correspond en réalité à un arbre dont la racine est le concept que l'on cherche à décrire et qui constitue le point de départ de la navigation dans la famille

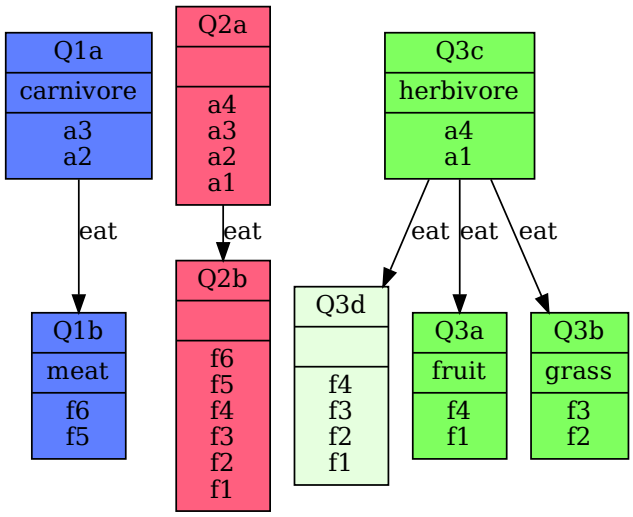

Figure 8.3 – Ensemble de graph patterns de CG_3.

de treillis. Comme on peut le constater, cette notion de "graphe navigation" se rapproche de celle de *Projected Graph Pattern* (PGP) qui simule un concept RCA (définition 7.4). Il suffit d'abstraire les extensions des concepts à l'aide des variables et d'étiqueter les arêtes par les noms de relations des attributs relationnels (le quantificateur peut être omis, puisque nous ne considérons que le quantificateur $\exists$). On obtient ainsi un graphe orienté et étiqueté qui représente la description du nœud racine. À titre d'illustration, le graphe de la figure 8.5, exprimant que *les herbivores consomment des fruits et des herbes*, constitue une version généralisée du graphe navigation de la figure 8.4, et capture uniquement l'information intensionnelle.

8.1.2 Intension d'un concept GCA

GCA produit en sortie un ensemble de *graph patterns*, chacun représentant un ensemble maximal de concepts interdépendants pour l'interprétation de leurs intensions respectives. Comme décrit à la section 4.5.2.1, l'intension de chaque concept d'un graph pattern s'obtient par projection du pattern sur le concept (nœud) concerné. Par définition, l'intension d'un concept Qix dans un pattern P, est donnée par le PGP $((x), Pi_r)$, où Pi_r correspond au sous-pattern constitué du nœud x et du pattern core de P.

À titre d'illustration, considérons le concept $Q3c$ du pattern Q3 en vert (figure 8.3). Le pattern core de Q3 correspond au sous-pattern (en couleur vive) constitué des concepts $Q3c$, $Q3a$ et $Q3b$. Ainsi, seuls les concepts du pattern core sont impliqués dans la descrip-

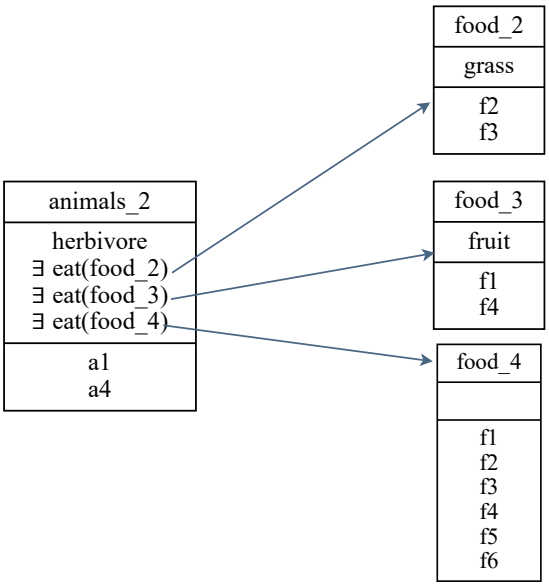

FIGURE 8.4 – Graphe représentant les concepts impliqués dans l'intension de $animals\_2$.

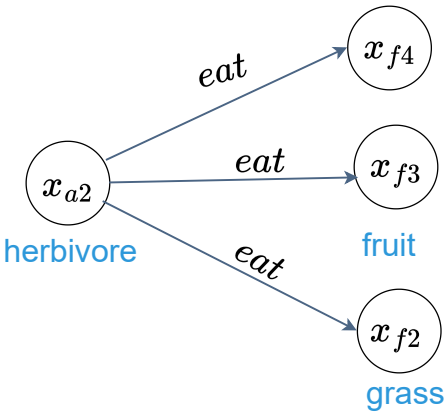

FIGURE 8.5 – Graph d'intension correspondant au graphe de description de la figure 8.4.

tion de l'intension de $Q3c$, car $Q3c$ fait lui-même partie du pattern core. La figure 8.6 met en évidence la projection du pattern Q3 sur le concept $Q3c$ au sens de la définition de l'intension d'un concept GCA. Dans cette figure, le sous-pattern encadré illustre l'ensemble des concepts qui participent à la description de $Q3c$ tout en capturant les différentes connexions entre ces concepts. Ce sous-pattern (encadré) peut également être interprété comme un arbre ayant pour racine le tuple de projection $Q3c$.

En comparant les illustrations des figures 8.4 et 8.6, on observe que la définition du PGP

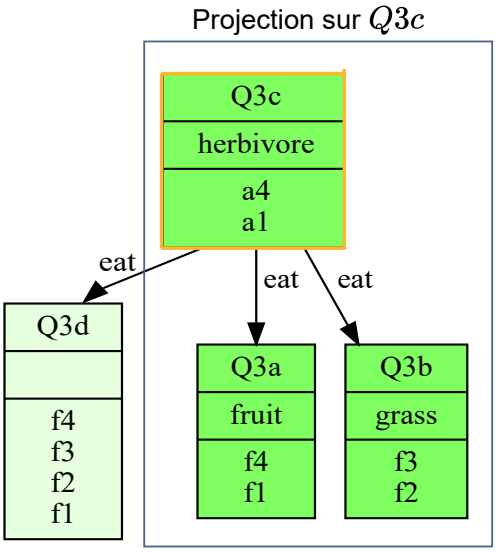

Figure 8.6 – Projection du pattern Q3 sur le concept $Q3c$.

dans GCA permet de recentrer le pattern sur le concept à analyser tout en éliminant les redondances. Le concept $Q3d$ généralise les concepts $Q3a$ et $Q3b$ en regroupant l'ensemble des aliments consommés par les animaux du concept $Q3c$. Cependant, par définition, GCA exclut $Q3d$ de l'intension de $Q3c$, le considérant comme redondant pour sa description. En revanche, du côté de RCA où l'intension complète d'un concept s'obtient par héritage descendant des attributs, tous les attributs relationnels d'un concept sont pris en compte pour l'interprétation de son intension.

8.2 Des familles de treillis de concepts RCA aux patterns relationnels

Dans cette section, nous décrivons le processus de transformation d'une famille de treillis de concepts RCA en un ensemble de patterns relationnels semblables aux graph patterns de GCA. Ce processus de transformation s'appuie sur les notions de graphe de dépendance et de graphe de concepts d'une famille de treillis, telles que définies dans [Ferré et Cellier, 2018].

8.2.1 Graphe de dépendance d'une famille de treillis de concepts

La transformation d'une famille de treillis de concepts en une hiérarchie de graphes de concepts a été décrite dans [Ferré et Cellier, 2018] sur la base des composantes fortement

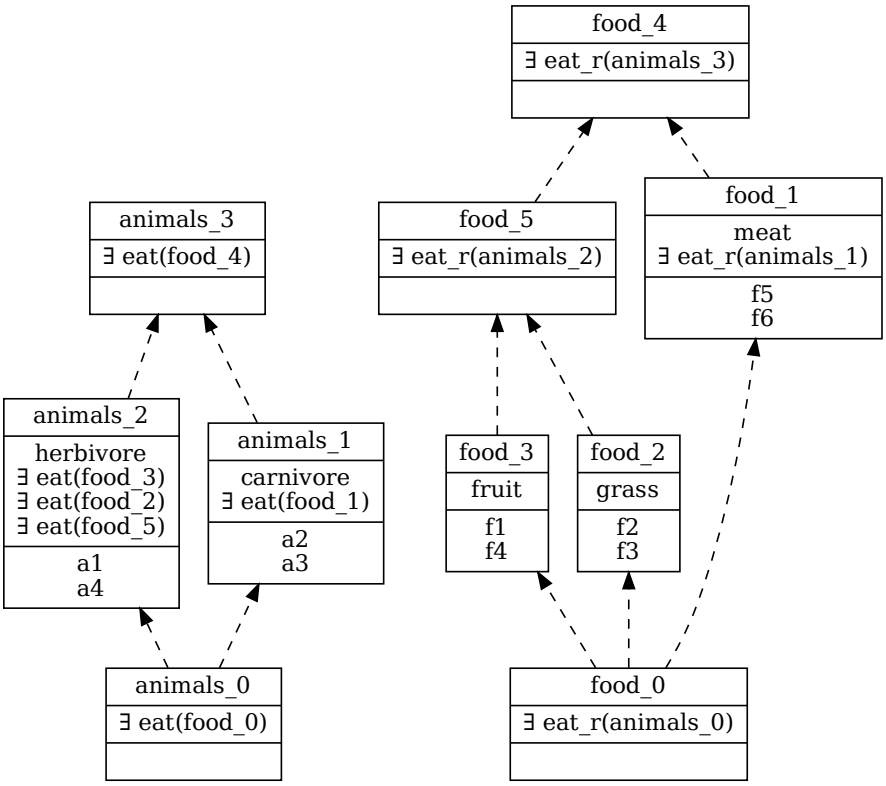

FIGURE 8.7 – Résultats RCA sur FRC_3_r qui intègre la relation inverse *eat_r*.

connexes (*Strongly Connected Components* - SCC) [EVEN, 2011] du *graphe de dépendance* (définition 5.1) entre les concepts. Comme décrit dans [FERRÉ et CELLIER, 2018] et tel que présenté dans le section 5.3.2, l'intuition sous-jacente au graphe de dépendance est que l'intension d'un concept dépend de ses ancêtres dans le treillis (dépendances intra-treillis, induites par la relation de subsumption) et des concepts cibles de ses attributs relationnels (dépendances inter-treillis, induites par les attributs relationnels).

Nous illustrons les définitions de graphe de dépendance (définition 5.1) et de graphe de concepts (définition 5.2) à l'aide de l'exemple FRC_3 du tableau 8.1. La figure 8.7 montre la famille de treillis RCA obtenue à partir de FRC_3_r, qui intègre les relations inverses (ici *eat_r*) à FRC_3. Le graphe de dépendance est ensuite construit sur cette famille de treillis de la figure 8.7. On dit qu'un concept C_1 dépend d'un concept C_2 (noté $C_1 \rightarrow C_2$), lorsque C_2 est un parent de C_1 ($C_1 \preceq C_2$), ou lorsque C_1 est étiqueté par un attribut relationnel $\exists r(C_2)$, c'est-à-dire $\exists r(C_2) \in Int(C_1)$. À titre d'illustration, dans la famille de treillis présentée à la figure 8.7 voici deux exemples de relations de dépendance entre concepts :

— $animals\_2 \rightarrow animals\_3$, car $animals\_2 \preceq animals\_3$

— $animals\_2 \rightarrow food\_3$, car $\exists eat(food\_3) \in Int(animals\_2)$.

La figure 8.8 montre le graphe de dépendance de la famille de treillis présentée à la figure 8.7 avec ses 3 SCCs mis en évidence par des cercles. Pour une meilleure lisibilité, les arêtes sont représentées de deux manières : les arêtes en pointillés représentent la relation de subsomption entre les concepts, tandis que les arêtes pleines et étiquetées matérialisent les attributs relationnels entre les concepts. Par exemple, l'arête entre les concepts $animals\_2$ et $food\_3$ étiquetée par $\exists$ *eat* matérialise le fait que $\exists$ *eat(food_3)* $\in$ *Int*$(animals\_2)$.

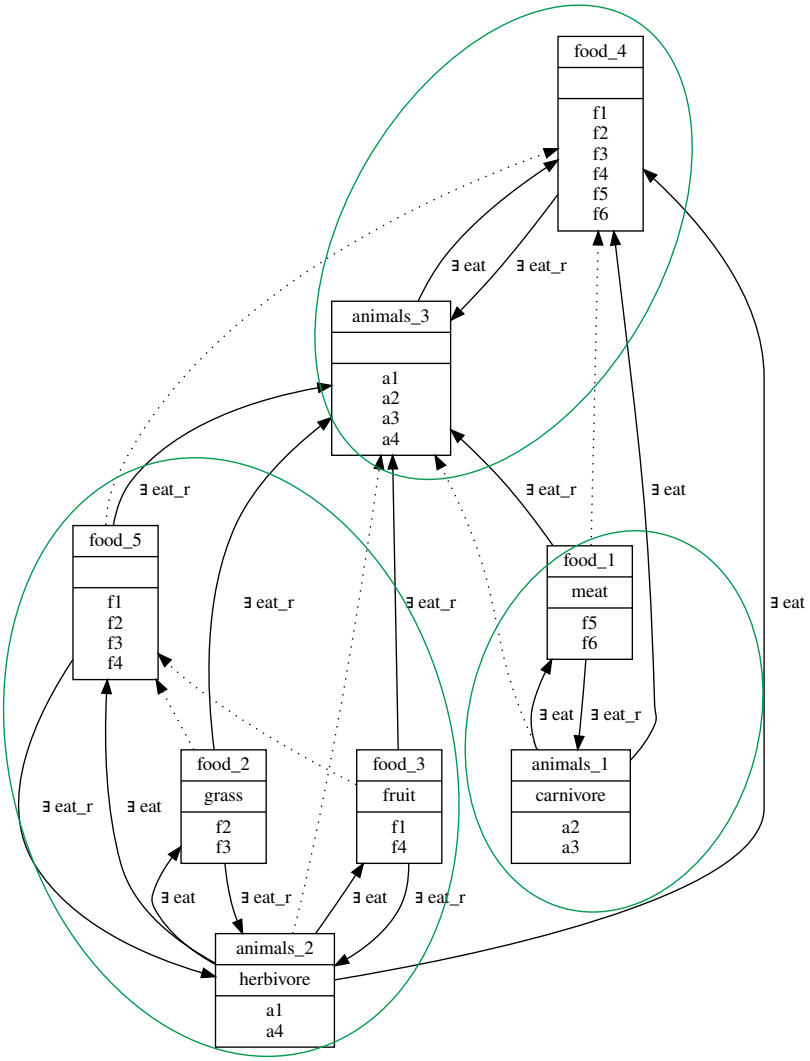

Figure 8.8 – Graphe de dépendance de la famille de treillis de la figure 8.7.

Un graphe de concepts est défini comme un SCC du graphe de dépendance, c'est-à-dire un ensemble maximal de concepts où chaque concept a un chemin de dépendance vers tous les autres concepts (du SCC). Pour capturer les structures relationnelles à partir de ces graphes de concepts, nous ne conserverons que les arêtes correspondant aux attributs

relationnels. En effet, on sait comment reconstruire les arêtes traduisant la relation de subsomption. Il existe deux difficultés pour obtenir des patterns relationnels semblables aux graph patterns GCA à partir de ces graphes de concepts.

— La première difficulté concerne les arêtes inter-SCC (par exemple, l'arête $\exists$ *eat* entre les concepts $animals\_1$ et $food\_4$ dans la figure 8.8), la question est de savoir si on perd de l'information sur les intensions des concepts en ignorant ces arêtes inter-SCC.
— La deuxième difficulté réside dans la présence de relations inverses qui ne sont pas explicitement représentées dans les graph patterns GCA.

Il convient de noter que les arêtes inter-SCC n'ont pas d'arêtes réciproques, sinon elles seraient des arêtes intra-SCC. Ces deux difficultés sont résolues dans la section suivante par la notion de redondance des attributs relationnels.

8.2.2 Attributs relationnels redondants

Une interprétation naïve de l'intension d'un concept prend en compte tous ses attributs relationnels, dont certains n'apportent aucune nouvelle information sur la description du concept. Par exemple, l'intension complète du concept $animals\_2$ (figure 8.7) contient les attributs relationnels $\exists$*eat*($food\_2$), $\exists$*eat*($food\_3$), $\exists$*eat*($food\_5$) et $\exists$*eat*($food\_4$). Les deux premiers attributs relationnels suffisent pour la description du concept $animals\_2$, tandis que $\exists$*eat*($food\_5$) et $\exists$*eat*($food\_4$) sont redondants. En effet, ces deux derniers attributs n'apportent aucune information nouvelle par rapport à $\exists$*eat*($food\_2$) et $\exists$*eat*($food\_3$), car le concept $food\_4$ subsume le concept $food\_5$ qui est lui-même le subsumant direct des concepts $food\_2$ et $food\_3$. Le lemme 3 introduit la notion de redondance des attributs relationnels, telle qu'elle est définie dans [Nica et al., 2016a ; Nica et al., 2016c].

Lemme 3. Soient C_1 et C_2 deux concepts tels que $C_1 \preceq C_2$. Si un concept C est tel que $\exists\, r(C_1) \in Int(C)$, alors nous avons également $\exists\, r(C_2) \in Int(C)$. Pour cette raison, $\exists\, r(C_2)$ est considéré comme redondant pour la description de C.

Démonstration. Soient deux concepts $C_1 = (X_1, Y_1)$ et $C_2 = (X_2, Y_2)$.
$\exists r(C_1) \in Int(C) \iff \forall o \in Ext(C), r(o) \cap X_1 \neq \varnothing$. Puisque $C_1 \preceq C_2$, $X_1 \subseteq X_2$ et donc, $r(o) \cap X_2 \neq \varnothing \iff \exists r(C_2) \in Int(C)$. Par conséquent, les attributs relationnels sont ordonnés et $\exists r(C_2)$ est redondant dans l'interprétation de C. □

Ce lemme définit les redondances sur la base desquelles on peut réduire la description d'un concept à son intension la plus spécifique. Cependant, en examinant plus en détail l'exemple du concept $animals\_2$, nous pouvons différencier les deux attributs redondants $\exists$ *eat*($food\_5$) et $\exists$ *eat*($food\_4$). Le premier fait référence au subsumant direct des deux concepts $food\_2$ et $food\_3$, tandis que le second fait référence au concept $food\_4$ qui couvre également le concept $food\_1$ et contient des objets supplémentaires ($f5, f6$) qui

n'ont aucun lien avec les objets de $animals\_2$ que l'on cherche à décrire. Ceci ajoute une sorte de bruit à la description de $animals\_2$. Logiquement, il n'y a pas d'attribut relationnel inverse dans l'intension de $food\_4$ qui pointe vers $animals\_2$, car $food\_4$ contient des objets n'ayant aucun lien avec ceux de $animals\_2$. Ce type d'attribut est qualifié de *fortement redondant* et est défini dans le lemme suivant.

Lemme 4. Soient $C = (X, Y)$ et $C_1 = (X_1, Y_1)$ deux concepts formels. Si $\exists\, r(C_1) \in Y$ et $\exists\, r^-(C) \notin Y_1$, alors il existe un concept $C_2 = (X_2, Y_2) \preceq C_1$ tel que $\exists\, r(C_2) \in Y$ et $\exists\, r^-(C) \in Y_2$ avec $X_2 = X_1 \cap \bigcup\{r(o) \mid o \in X\}$. On dit alors que $\exists\, r(C_1)$ est fortement redondant pour la description de C.

Démonstration. Nous devons prouver que : (1) $X_2 \subseteq X_1$, ce qui est trivial par définition de X_2, (2) $\exists r^-(C) \in Y_2$, et (3) $\exists r(C_2) \in Y$.

Pour prouver que $\exists r^-(C) \in Y_2$, nous partons de ce que cela implique d'être un élément de $X_2 : \forall o_j \in X_2 : o_j \in \cup\{r(o) \mid o \in X\}$, par définition de X_2
$\Rightarrow \forall o_j \in X_2 : \exists o_i \in X : (o_i, o_j) \in r$, par définition de $r(o_i)$
$\Rightarrow \forall o_j \in X_2 : \exists o_i \in X : (o_j, o_i) \in r^-$, par définition de la relation inverse
$\Rightarrow \exists r^-(C) \in Y_2$, par définition des attributs relationnels dans l'intension de C_2.

Pour prouver que $\exists r(C_2) \in Y$, nous partons du fait que $\exists r(C_1) \in Y$, ce qui implique
$\forall o_i \in X : \exists o_j \in X_1 : (o_i, o_j) \in r$
$\Rightarrow \forall o_i \in X : \exists o_j \in X_1 : (o_i, o_j) \in r$, donc $o_j \in \cup\{r(o_i) \mid o_i \in X\}$
$\Rightarrow \forall o_i \in X : \exists o_j \in X_2 : (o_i, o_j) \in r$, par définition de X_2
$\Rightarrow \exists r(C_2) \in Y$, par définition des attributs relationnels dans l'intension de C. □

Le lemme 4 établit qu'un attribut relationnel ne possédant pas d'attribut relationnel *réciproque* est fortement redondant ; c'est précisément le cas des arêtes inter-SCC. Par conséquent, ignorer les arêtes inter-SCC n'entraîne pas de perte d'information dans l'interprétation des intensions de concepts. Cela permet ainsi de résoudre la première difficulté relative aux arêtes inter-SCC, comme mentionné dans la section précédente. Une fois supprimés les attributs relationnels qualifiés de fortement redondants, tous les attributs relationnels restants possèdent un attribut inverse (ou réciproque). Cela signifie que, les arêtes induites par les relations inverses peuvent être ignorées à l'instar de ce qui est fait dans les graph patterns GCA. Cette simplification répond à la deuxième difficulté, liée à la présence des attributs relationnels inverses dans les patterns relationnels.

8.2.3 Patterns relationnels RCA

Un pattern relationnel RCA est induit par chaque SCC du graphe de dépendance de la famille de treillis de concepts. Cette section définit d'abord un tel pattern relationnel comme

un pattern d'extensions, d'attributs et de relations (*Extent-Attribute-Relation pattern (EAR-pattern)*, en anglais), qui servira de *représentation commune* pour comparer les intensions des concepts RCA et GCA. Ensuite, les EAR-patterns de RCA (définition 8.2) sont définis sur la base de ce qui précède, c'est-à-dire en utilisant uniquement les arêtes relationnelles intra-SCC et en ignorant les arêtes inverses en raison de leur redondance.

Définition 8.1 (EAR-pattern). Soit O un ensemble d'objets, A un ensemble d'attributs (formels) et R un ensemble de relations binaires. Un *Extent-Attribute-Relation pattern* (EAR-pattern) est un graphe orienté étiqueté $P = (V, E, L_V, L_E)$ où :

- les nœuds $V \subseteq 2^O$ sont les extensions de concepts, c'est-à-dire des ensembles d'objets ;
- $E \subseteq V \times V$ est un ensemble d'arêtes orientées reliant les extensions ;
- $L_V : V \to 2^A$ est la fonction d'étiquetage des nœuds par des ensembles d'attributs formels ;
- $L_E : E \to 2^R$ est la fonction d'étiquetage des arêtes par des ensembles de relations.

Définition 8.2 (EAR-pattern RCA). Soit S un SCC d'un graphe de dépendances $G_{\mathbf{L}}$. Le *EAR-pattern RCA* induit par S est le EAR-pattern $P_S = (V, E, L_V, L_E)$ où :

- $V = \{Ext(C) \mid C \in S\}$: extensions des concepts de S ;
- $E = \{(Ext(C_1), Ext(C_2)) \mid C_1, C_2 \in S, \exists r(C_2) \in Int(C_1)\}$: les arêtes correspondent aux attributs relationnels dans les intensions des concepts ;
- $L_V(Ext(C)) = \{a \in A \mid a \in Int(C)\}$: les étiquettes des extensions sont les attributs unaires dans les intensions des concepts ;
- $L_E((Ext(C_1), Ext(C_2))) = \{r \in R \mid \exists r(C_2) \in C_1\}$: les étiquettes des arêtes sont les noms de relations des attributs relationnels reliant les concepts.

Pour illustration, la figure 8.9 présente l'ensemble des EAR-patterns RCA obtenus à partir de la famille de treillis illustrée à la figure 8.7. Comme on peut le constater, cet ensemble d'EAR-patterns RCA correspond à l'ensemble de graph patterns GCA obtenu sur le même exemple, tel que présenté à la figure 8.3. Il est important de souligner que pour une famille de treillis de concepts donnée, l'ensemble des EAR-patterns RCA correspondant est équivalent en termes d'informations à la famille de treillis de concepts d'origine.

En résumé, pour une famille de treillis de concepts obtenue à partir d'une famille relationnelle de contextes (intégrant les relations inverses), l'ensemble des EAR-patterns RCA correspondants est construit selon le processus suivant : (1) construction du graphe de dépendance de la famille de treillis, (2) calcul des SCCs du graphe de dépendance, (3) suppression des arêtes induites par la relation de subsomption, (4) suppression des attributs relationnels *fortement* redondants et (5) suppression des attributs relationnels induits par les relations inverses. Il convient de noter que ces résultats sont obtenus grâce aux relations inverses qui jouent un rôle important dans le calcul des SCCs.

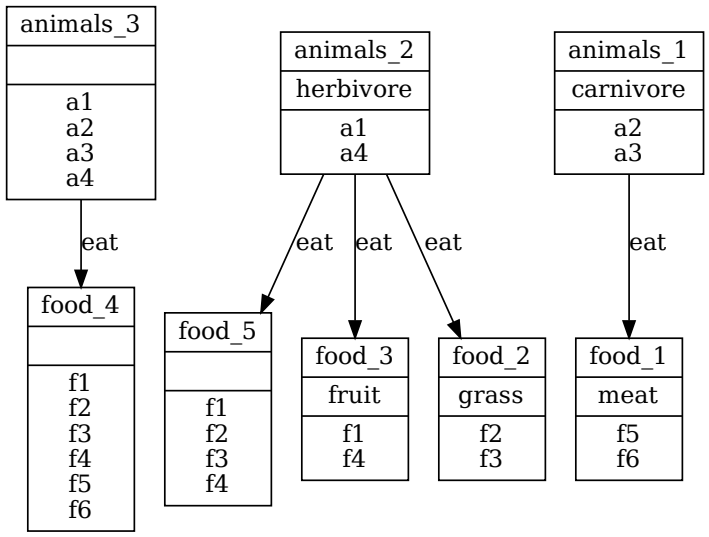

Figure 8.9 – EAR-patterns RCA de la famille de treillis présentée à la figure 8.7.

Grâce à cette transformation des résultats de RCA sous forme de EAR-patterns, comparables aux graph patterns de GCA, il devient possible d'envisager une comparaison entre les deux types de patterns. Cependant, les EAR-patterns RCA et les graph patterns GCA ne sont pas toujours directement comparables, car les graph patterns GCA peuvent souvent comporter plusieurs nœuds correspondant au même concept, ce que l'on désigne par les *concepts automorphes*. Par conséquent, nous commençons par transformer les graph patterns GCA en EAR-patterns.

8.3 Des graph patterns GCA aux EAR-patterns

En GCA, la duplication des concepts est parfois nécessaire pour représenter correctement les graph patterns présentant des symétries ou capturant certaines structures, telles que les cycles. Les concepts dupliqués, appelés *concepts automorphes*, représentent différentes occurrences d'un même concept, comme décrit à la section 4.6 (chapitre 4). En revanche, RCA ne produit pas de tels concepts automorphes. L'approche consiste donc à fusionner les occurrences de concepts automorphes dans les graph patterns GCA. Pour un graph pattern P, fusionner ses concepts automorphes revient à calculer son *graphe quotient* par rapport à ses extensions de concepts, c'est-à-dire que deux nœuds sont regroupés dans la même classe s'ils partagent la même extension.

Définition 8.3 (Graphe quotient). Soit G un graphe et soit $\mathcal{P} = V_1, \ldots, V_k$ une partition

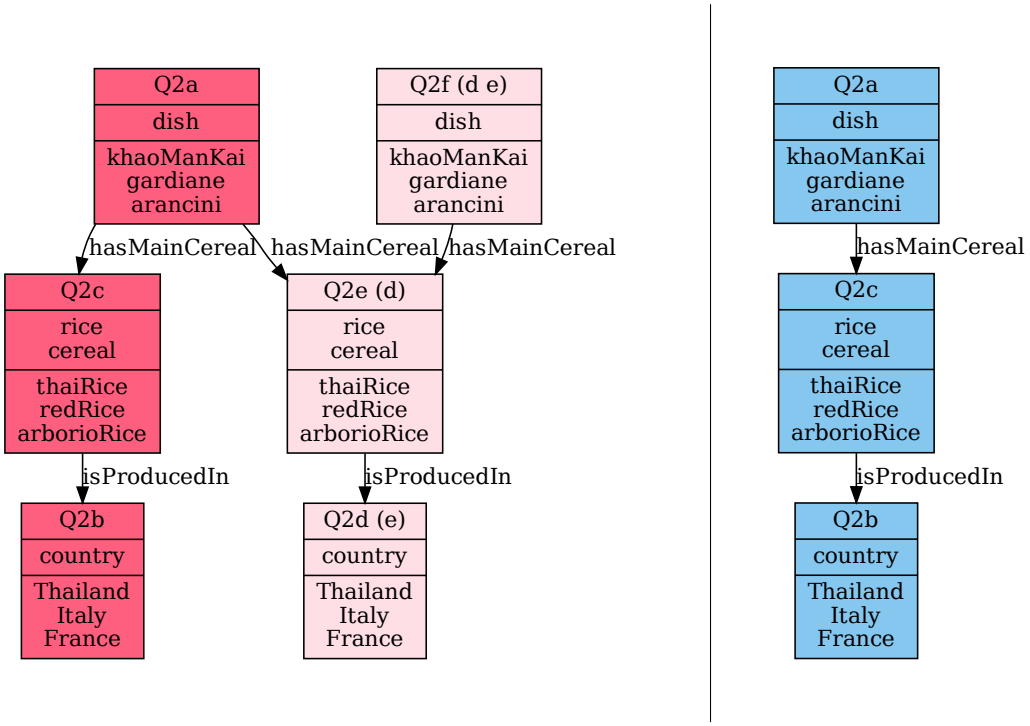

FIGURE 8.10 – Graph pattern Q2 avec concepts automorphes (à gauche) et une représentation de son graphe quotient correspondant à droite.

de l'ensemble des sommets de G en classes non vides. Le *quotient* $G/\mathcal{P}$ de G par $\mathcal{P}$ est le graphe dont les sommets sont les ensembles $V_1, \ldots, V_k$ et dont les arêtes sont les paires $(V_i, V_j), i \neq j$, telles qu'il existe $u_i \in V_i$, $u_j \in V_j$ et $(u_i, u_j) \in E(G)$ [HAHN et TARDIF, 1997].

Pour illustration, la figure 8.10 présente le graph pattern Q2 (issu de l'exemple traité à la section 4.6) ainsi qu'une représentation de son graphe (pattern) quotient. Dans ce pattern quotient, chacune des deux occurrences d'un même nœud de Q2 est fusionnée en une seule occurrence.

Dans un *pattern quotient* $P \subseteq \mathcal{V}^* \times A$ d'un graph pattern, A contient des attributs unaires qui étiquettent les nœuds (identifiés par des ensembles d'objets) et contient les attributs n-aires (relations) qui étiquettent les arêtes connectant n nœuds. Si P est limité aux relations binaires, comme c'est le cas dans ce travail, A peut être partitionné en deux ensembles : les attributs unaires (A_1) et les attributs binaires (A_2), également appelés relations binaires. Le pattern P peut donc être représenté de manière équivalente comme un EAR-pattern selon la définition suivante.

Définition 8.4 (EAR-pattern GCA). Soit $P \subseteq \mathcal{V}^* \times A$ le graphe quotient d'un graph pattern GCA. Soit $A = A_1 \cup A_2$, avec A_1 l'ensemble des attributs unaires et A_2 l'ensemble des attributs binaires. Le EAR-pattern de P s'écrit $P_G = (V, E, L_V, L_E)$ où :

- $V = \{v \mid \exists a \in A, ((\ldots, v, \ldots), a) \in P\}$;
- $E = \{(v_1, v_2) \mid \exists a \in A, ((v_1, v_2), a) \in P\}$;
- $L_V(v) = \{a \in A_1 \mid ((v), a) \in P\}$;
- $L_E((v_1, v_2)) = \{r \in A_2 \mid ((v_1, v_2), r) \in P\}$.

Compte tenu de ces définitions, les résultats produits par RCA et GCA peuvent être transformés dans une représentation commune, appelée EAR-patterns, permettant ainsi la comparaison de leurs intensions de concepts. La section suivante démontre que l'ensemble des intensions de concepts issus de RCA est inclus dans celui de GCA.

8.4 Comparaison des EAR-Patterns RCA et GCA

Soient $(\mathbf{K}, \mathbf{R})$ une FRC, K le contexte graphe correspondant, et $F = (\mathbf{K}, \mathbf{R} \cup \mathbf{R}^-)$ la FRC étendue avec les relations inverses. Soit $S = \bigcup\{S_1, S_2, ..., S_m\} = (V_S, E_S, L_{V_S}, L_{E_S})$ l'union disjointe[1] des EAR-patterns RCA de F et soit $P = \bigcup\{P_1, P_2, ..., P_n\} = (V_P, E_P, L_{V_P}, L_{E_P})$ l'union disjointe des EAR-patterns GCA de K. Dans ce qui suit, nous prouvons que S est un *sous-graphe* de P au sens de la définition 8.5, et, par conséquent, que chaque EAR-pattern RCA est un *sous-graphe* d'un EAR-pattern GCA.

Définition 8.5 (Sous-graphe). Soient $G_1 = (V_1, E_1, L_{V1}, L_{E1})$ et $G_2 = (V_2, E_2, L_{V2}, L_{E2})$ deux graphes étiquetés. G_2 est un sous-graphe étiqueté de G_1 ($G_2 \subseteq G_1$) si et seulement si $V_2 \subseteq V_1, E_2 \subseteq E_1, \forall v \in V_2 : L_{V_2}(v) = L_{V_1(v)}$ et $\forall (v_1, v_2) \in E_2 : L_{E_2}((v_1, v_2)) = L_{E_1}((v_1, v_2))$.

Avant d'énoncer notre théorème, nous introduisons le lemme 5, qui servira pour sa démonstration.

Lemme 5. Pour tout $R \in \mathcal{R}_k$, and $\pi \in \Pi_k^l$, nous avons $\pi(int(R)) = int(\pi(R))$ [Ferré et Cellier, 2020].

Démonstration. Soit $R = \{\overline{o}_1, \ldots, \overline{o}_n\}$. $\pi(int(R)) = \pi(\bigcap_{\overline{o} \in R} (\overline{o}, I))$
$= \pi((\psi(\overline{o}_1, \ldots, \overline{o}_n), I \times \ldots \times I)) = (\pi(\psi(\overline{o}_1, \ldots, \overline{o}_n)), I \times \ldots \times I)$
$= (\psi(\pi(\overline{o}_1), \ldots, \pi(\overline{o}_n)), I \times \ldots \times I) = \bigcap_{\overline{o} \in R} (\pi(\overline{o}), I) = \bigcap_{\overline{o} \in \pi(R)} (\overline{o}, I) = int(\pi(R))$ □

Théorème 4. Soit $F = (\mathbf{K}, \mathbf{R} \cup \mathbf{R}^-)$ une FRC étendue avec des relations inverses et K le contexte graphe correspondant à la FRC $(\mathbf{K}, \mathbf{R})$. Soit $S = \bigcup_i S_i$ l'union des EAR-patterns RCA de F et soit $P = \bigcup_j P_j$ l'union des EAR-patterns GCA de K. Nous avons $S \subseteq P$, et donc $\forall\, S_i : \exists P_j : S_i \subseteq P_j$.

1. Chaque concept appartient à un et un seul pattern.

Démonstration. Soit C_R l'ensemble des extensions de concepts RCA de F et C_G l'ensemble des extensions de concepts GCA de K. En nous basant sur le résultat $C_R \subseteq C_G$ (théorème 3), nous voulons prouver que $S \subseteq P$ en démontrant que :

— $\forall (v_1, v_2) \in E_S : (v_1, v_2) \in E_P, L_{E_S}((v_1, v_2)) = L_{E_P}((v_1, v_2))$ **(1)**

— et $\forall v_i \in V_S : v_i \in V_P, L_{V_S}(v_i) = L_{V_P}(v_i)$ **(2)**.

(1) Soit $(v_1, v_2) \in E_S$ une arête, et $r \in L_{E_S}(v_1, v_2)$ une relation dans son étiquette, où v_1 et v_2 correspondent aux extensions des concepts C_1, C_2 respectivement. Cela implique que $\exists r(C_2) \in Int(C_1)$ et $\exists r^-(C_1) \in Int(C_2)$, par définition des EAR-patterns RCA. Alors $\forall o_1 \in v_1 : \exists o_2 \in v_2 : (o_1, o_2) \in r$ et $\forall o_2 \in v_2 : \exists o_1 \in v_1 : (o_2, o_1) \in r^-$, donc $(o_1, o_2) \in r$ (*), d'après la définition des attributs relationnels.

Soit $R_{12} = \{(o_1, o_2) \mid o_1 \in v_1, o_2 \in v_2, (o_1, o_2) \in r\}$ un *object relation* (définition 4.6) d'arité 2. Nous avons $\pi_1(R_{12}) = \{\pi_1(o_1, o_2) \mid (o_1, o_2) \in R_{12}\} = v_1$ et $\pi_2(R_{12}) = \{\pi_2(o_1, o_2) \mid (o_1, o_2) \in R_{12}\} = v_2$ [2], selon (*).

Soit $Q_{12} = int(R_{12}) = \cap_q\{Q((o_1, o_2))\}_{(o_1,o_2)\in R_{12}} = ((y_1, y_2), P_{12})$, pour un certain pattern P_{12}, par définition de l'intension d'un *object relation* d'arité 2. Cela implique que $r(y_1, y_2) \in P_{12}$, par définition de R_{12}.

Prouvons que $ext(((y_1), P_{12})) = v_1$ et $ext(((y_2), P_{12})) = v_2$, c'est-à-dire que les nœuds y_1, y_2 dans le EAR-pattern GCA correspondent aux nœuds v_1, v_2 dans le EAR-pattern RCA.

$ext(((y_1), P_{12}))$
$= ext(\pi_1(Q_{12}))$, car $\pi_1((y_1, y_2)) = y_1$
$= ext(\pi_1(int(R_{12})))$, car Q_{12} est l'intension de R_{12}
$= ext(int(\pi_1(R_{12})))$, d'après le lemme 5
$= ext(int(v_1))$, puisque $\pi_1(R_{12}) = v_1$
$= v_1$, car $v_1 \in C_R \subseteq C_G$, donc v_1 est fermé en tant qu'une extension de concept. Par analogie, nous avons également $ext(((y_2), P_{12})) = v_2$. Ces résultats impliquent que $r(y_1, y_2) \in P_{12}$, donc, d'après la définition des EAR-patterns GCA, (v_1, v_2) est une arête dans P, et r est l'une de ses étiquettes.

De plus, si r est une étiquette de (v_1, v_2) dans P, alors $r(y_1, y_2) \in P_{12}$, donc on a $r(o_1, o_2)$ pour tous les $(o_1, o_2) \in ext(Q_{12})$. Par conséquent, pour tout $o_1 \in \pi_1(ext(Q_{12})) = v_1$, il existe $o_2 \in \pi_2(ext(Q_{12})) = v_2$ tel que $(o_1, o_2) \in r$. Il en résulte que $\exists r(C_2) \in Int(C_1)$. De même, nous avons $\exists r^-(C_1) \in Int(C_2)$. Par conséquent, r est une étiquette de l'arête (v_1, v_2) dans S.

(2) Soit $v \in V_S$ un nœud et $a \in L_{V_S}(v)$ un attribut d'étiquette, où v correspond à l'exten-

2. π_1 et π_2 représentent respectivement la projection sur la première et la deuxième composante d'un tuple ou d'un ensemble de tuples.

sion d'un concept C. Cela implique que $a \in Int(C)$ par définition des EAR-patterns RCA, donc $\forall o \in v : a(o)$.

Soit $Q = int(v) = ((y), P)$ pour un certain pattern P. Nous avons $a(y) \in P$ car Q généralise tous les objets de v, qui ont tous l'attribut a. De plus, $ext(Q) = ext(int(v)) = v$ car $v \in C_R \subseteq C_G$. Alors, d'après la définition des EAR-patterns GCA, v est un nœud de P, et a est l'une de ses étiquettes.

De plus, si a est une étiquette de v, alors $a(y) \in P$, donc on a $a(o)$ pour tout $o \in v$, d'où $a \in Int(C)$, et enfin a est une étiquette de v dans S.

□

Le théorème 4 couvre trois configurations. Premièrement, il peut arriver que $S = P$, c'est-à-dire que l'ensemble des EAR-patterns soit identique dans les résultats des deux approches. Un exemple illustratif est celui des animaux et des aliments qu'ils consomment : les figures 8.9 et 8.3 présentent respectivement l'ensemble des EAR-patterns RCA et l'ensemble graph patterns GCA (qui coïncide ici avec l'ensemble des EAR-patterns GCA, en l'absence de concepts automorphes).

Deuxièmement, il peut exister une correspondance biunivoque entre les EAR-patterns des deux approches, mais au moins une paire de patterns distincts S_i et P_j vérifie $S_i \subset P_j$. Les exemples concernant les voitures et les garages, présentés à la section 7.3.3, illustrent bien cette configuration où GCA produit des concepts supplémentaires par rapport à RCA. À titre d'illustration, la figure 8.11 présente le EAR-pattern GCA (P_2) incluant le concept $Q2z$ (en vert), absent du résultat de RCA pour l'exemple CG_{1a}/ FRC_{1a}_r. La figure 8.12 présente le EAR-pattern RCA (S_3) correspondant. Ces deux figures mettent en évidence la relation $S_3 \subset P_2$.

La troisième configuration apparaît lorsqu'il existe un pattern P_j qui ne peut être associé à aucun pattern S_i. C'est notamment le cas de l'exemple portant sur les cycles, qui sera présenté et analysé dans le chapitre 9.

En conclusion, chaque EAR-pattern de RCA est un sous-graphe d'un EAR-pattern de GCA. Par conséquent, l'ensemble des intensions de concepts RCA est inclus dans celui des concepts GCA.

8.5 Potentiel pratique des EAR-patterns RCA

Au-delà de leur rôle de passerelle pour la comparaison intensionnelle entre RCA et GCA, les EAR-patterns RCA constituent une base solide pour faciliter l'interprétation de la famille de treillis RCA. Ils offrent une représentation compacte et lisible des résultats de RCA, avec suppression des redondances. Ainsi, les EAR-patterns améliorent à la fois l'exploration et

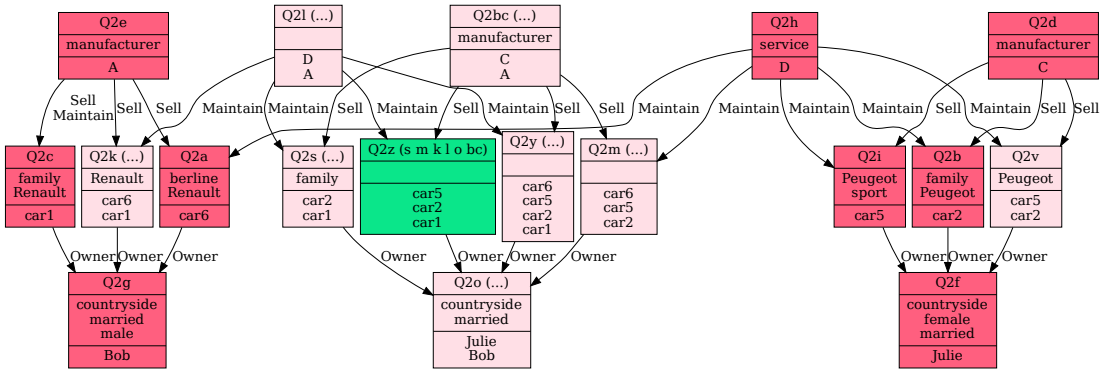

FIGURE 8.11 – Représentation du EAR-pattern GCA contenant le concept $Q2z$ non produit par RCA. La notation parenthésée des noms des concepts est abrégée par "(...)" pour plus lisibilité. Par exemple, dans $Q2l\,(\ldots)$, la partie "(...)" représente les éléments $(s\ m\ k\ o\ bc)$.

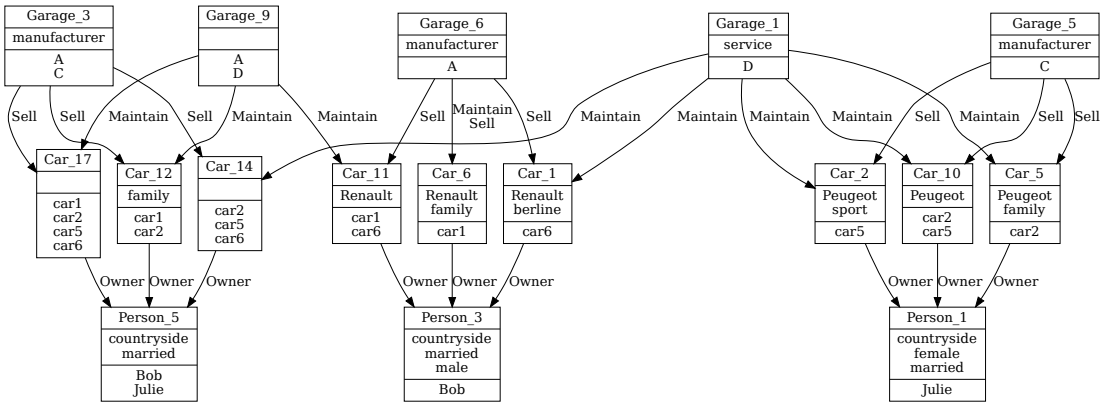

FIGURE 8.12 – EAR-pattern RCA associé au EAR-pattern GCA de la figure 8.11.

l'interprétation des résultats en proposant une vue synthétique et structurée des relations inter-concepts.

À l'instar du travail sur la représentation de la famille de treillis RCA sous forme d'une hiérarchie de graphes de concepts [FERRÉ et CELLIER, 2018], il est possible de structurer hiérarchiquement l'ensemble des EAR-patterns RCA. Une telle hiérarchie pourrait être construite en se basant sur la relation de subsomption entre les concepts (du point de vue des extensions), offrant ainsi une représentation plus lisible que celle proposée dans [FERRÉ et CELLIER, 2018], dans la mesure où les attributs relationnels strictement redondants ont été supprimés dans les EAR-patterns RCA.

La transformation des résultats de RCA en un ensemble de EAR-patterns, constitue également une base solide pour un usage combiné des approches RCA et GCA. Par exemple, alors que GCA utilise les concepts automorphes pour capturer des structures symétriques ou la présence des cycles dans les données, les EAR-patterns RCA permettent de représenter des structures plus générales, faisant abstraction des détails spécifiques capturés par GCA via les concepts automorphes. Une analyse conjointe de ces deux types de structures offrirait la possibilité d'extraire différents niveaux de connaissances.

De plus, les EAR-patterns RCA, tels qu'ils sont définis actuellement avec le quantificateur existentiel $\exists$, fournissent une base qu'on peut étendre à des familles de treillis construites avec d'autres quantificateurs de *scaling* ($\exists\forall$, $\exists \supseteq$, etc.) [Braud et al., 2018]. Cela permettrait de dépasser la limitation actuelle de GCA, dont les patterns n'utilisent implicitement que le quantificateur $\exists$, et de produire des patterns relationnels plus représentatifs des structures présentes dans les données. L'intuition serait alors de redéfinir et d'adapter la notion de redondance des attributs relationnels pour différents quantificateurs.

8.6 Conclusion

Dans chapitre, nous avons conduit une étude comparative de RCA et de GCA du point de vue de leurs intensions de concepts dans leur cadre commun. Celui-ci inclut notamment l'utilisation du quantificateur existentiel $\exists$, le calcul des concepts unaires, le traitement des relations binaires, ainsi que l'intégration des relations inverses dans les données de RCA.

Pour mener à bien cette comparaison, nous avons transformé la famille de treillis RCA en un ensemble de patterns relationnels semblables aux graph patterns GCA, afin de rendre les intensions de concepts des deux approches directement comparables. Nous avons ensuite établi que l'ensemble des intensions de concepts RCA est inclus dans celui de GCA, en démontrant que chaque pattern relationnel RCA constitue un sous-graphe d'un pattern GCA pour un même jeu de données. Ce résultat, combiné à celui de la comparaison extensionnelle – qui démontre que l'ensemble des extensions de concepts RCA est inclus dans celui de GCA – permet de conclure que les résultats de RCA sont inclus dans ceux GCA. Par conséquent, GCA apparaît comme plus expressif que RCA dans leur cadre commun.

La comparaison de RCA et GCA sur leurs dimensions extensionnelle et intensionnelle a été menée dans une optique de rapprochement des deux approches. Le chapitre suivant propose, une comparaison de RCA et GCA à travers leurs différences.

Chapitre 9

Analyse comparative de RCA et de GCA à travers leurs différences

Sommaire

Comme mentionné précédemment, RCA et GCA diffèrent sur plusieurs aspects, notamment la modélisation des relations n-aires et le traitement des cycles. Dans ce chapitre, nous proposons une analyse comparative de RCA et de GCA à travers leurs différences. Plutôt que de chercher à rapprocher les deux approches, l'objectif est d'examiner dans quelle mesure leurs divergences peuvent se révéler complémentaires et bénéfiques pour l'analyse. Nous abordons ainsi les différences liées à la modélisation des relations ternaires (et n-aires en général) ainsi qu'au traitement des cycles. La section 9.1 analyse l'influence de la modélisation des relations n-aires sur les résultats des deux approches, tandis que la section 9.2 met en évidence leurs différences dans le traitement des cycles. La section 9.3 présente une expérimentation de RCA et GCA sur un jeu de données réel, et la section 9.4 termine par un aperçu des atouts et limites de ces deux approches.

9.1 Impacts de la modélisation des relations n-aires sur les résultats de RCA et de GCA

Dans de nombreux domaines réels, les données sont de nature multi-relationnelles, ce qui permet de représenter plusieurs types d'interactions entre les entités. La représentation et l'analyse de ces données sont nécessaires pour comprendre des phénomènes complexes impliquant de multiples entités. À titre d'exemple, dans le domaine de la recherche, un chercheur peut être affilié à plusieurs institutions, collaborer avec différents chercheurs et organiser des événements avec d'autres chercheurs. L'analyse de données multi-relationnelles peut nécessiter des choix de modélisation, tels que la représentation des relations n-aires, et ce choix peut avoir une incidence sur les réponses aux questions d'analyse.

En matière de traitement de relations, RCA se limite aux relations binaires, tandis que GCA est conçue pour prendre en charge directement les relations n-aires d'arité quelconque. Ainsi, les relations n-aires doivent être transformées en relations binaires afin de s'adapter au format de données de RCA [Keip et al., 2019]. Cette section propose une comparaison pratique entre RCA et GCA, centrée sur leurs différence dans le traitement des relations n-aires, en mettant un accent sur leur capacité à répondre aux questions d'analyse. Cette comparaison des deux approches a été publiée dans [Fokou et al., 2025a].

9.1.1 Données et questions d'analyse

Pour cette étude, nous considérons le contexte graphe CG_6 illustré à la figure 9.1, qui décrit un réseau de chercheurs, leurs affiliations ainsi que leurs activités, telles que l'organisation de conférences et la publication d'articles[1]. Ce contexte graphe est composé de 5 catégories d'objets :

- $researcher$: décrit les chercheurs $\{R1, \ldots, R6\}$ par leur nombre de publications, qui peut être inférieur à 10 (*n_pub_inf_10*) ou supérieur à 10 (*n_pub_sup_10*), et par leur nombre de citations, qui peut être inférieur à 20 (*n_cit_inf_20*) ou supérieur à 20 (*n_cit_sup_20*) ;
- $institution$ ($inst$) : décrit les institutions $\{I1, \ldots, I4\}$ par leur type (*university*, *company*) et par leur pays ($France$, $Romania$) ;
- $paper$: décrit les articles $\{P1, \ldots, P4\}$ par leur type (*journal*, *conference* (*cf*)) et par le fait qu'ils ont été cités (*cited*) ;
- $role$: représente les différents rôles {*g_chair*, *p_chair*, *l_organizer*} correspondant à *general chair*, *program chair* et *local organizer* respectivement ;
- $conference$ ($conf$) : représente les différentes conférences $\{ICCS, ICFCA, CLA\}$.

1. Il convient de noter que ce contexte graphe ne représente qu'un extrait de ce que pourrait être un tel graphe de données pour une communauté de chercheurs dans un domaine donné (par exemple, l'AFC).

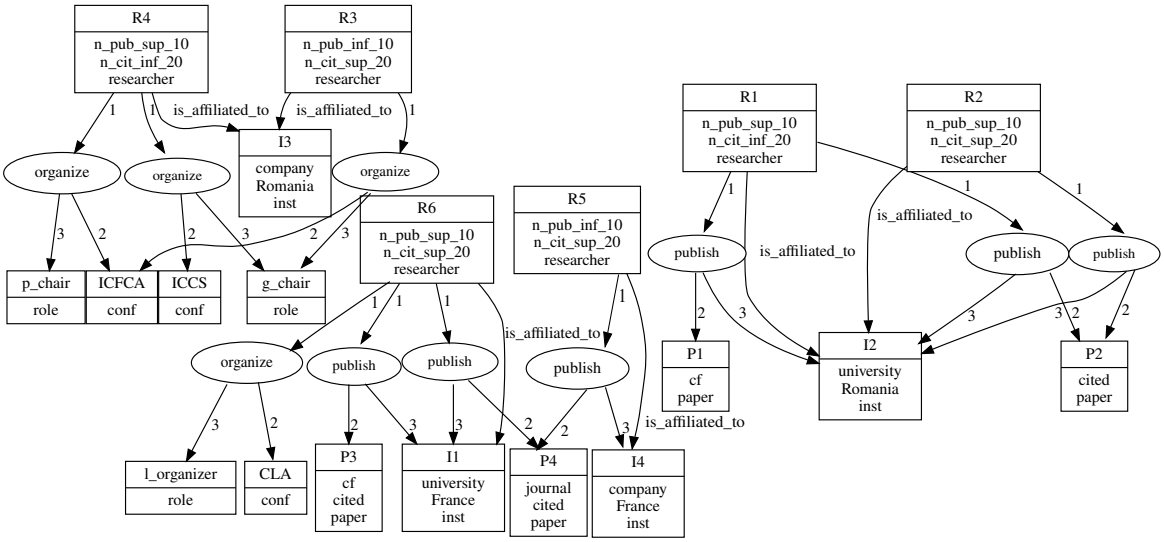

FIGURE 9.1 – Contexte graphe CG_6 décrivant les chercheurs et leurs activités.

Les catégories $role$ et $conference$ n'ont pas d'attributs. En ce qui concerne les relations, CG_6 comprend :

- la relation binaire *is_affiliated_to*$(researcher, inst)$, qui indique à quelle institution un chercheur est affilié ;
- la relation ternaire *publish*$(researcher, paper, inst)$, qui signifie qu'un chercheur publie un article avec une affiliation donnée ;
- et la relation ternaire *organize*$(researcher, conf, role)$, qui indique qu'un chercheur organise une conférence en y occupant un rôle donné.

GCA utilise des nœuds en ellipse pour représenter les relations n-aires (pour $n > 2$) et utilise des chiffres pour indiquer la position des entités dans la relation. Pour les relations ternaires, ces chiffres vont de 1 à 3, par exemple *organize*$(R4, ICFCA, p\_chair)$ dans la figure 9.1.

Sur un tel jeu de données, l'utilisateur peut formuler diverses requêtes ; en voici deux exemples :

- *query1* : l'ensemble des institutions dont les chercheurs ont organisé une conférence ;
- *query2* : l'ensemble des chercheurs qui ont publié des articles et sont affiliés à une institution française.

Nous nous appuyons sur ces deux requêtes pour examiner l'effet de la modélisation des données sur les résultats produits par RCA et GCA, et donc sur leurs réponses aux questions d'analyse. La section suivante présente la modélisation des relations ternaires dans les deux approches.

9.1.2 Modélisation des relations ternaires

RCA nécessite une modélisation initiale pour les relations n-aires, et cette modélisation n'est pas sans conséquences, car elle peut entraîner des pertes d'informations structurelles. La question qui se pose est de savoir comment modéliser efficacement les relations n-aires à la manière de RCA. En ce qui concerne GCA, la question porte sur la difficulté d'interpréter les patterns calculés directement sur les relations n-aires. Nous explorons comment chacune des approches RCA (suivant la modélisation des données) et GCA satisfait les questions *query1* et *query2*.

9.1.2.1 Modélisation des relations ternaires avec RCA

Pour répondre aux questions *query1* et *query2* avec RCA, nous recourons aux encodages par *réification, décomposition* et *partitionnement* présentés dans [Keip et al., 2020, 2019] pour modéliser les relations ternaires, comme décrit à la section 5.4. Parmi ces encodages, la réification n'entraîne pas de perte d'informations, la décomposition quant à elle peut conduire à une perte d'informations structurelles, tandis que le partitionnement entraîne une perte de connectivité.

Encodage par réification. Les relations *publish* et *organize* sont réifiées par la création de nouveaux contextes formels et relationnels. La relation *organize*$(researcher, conf, role)$ est encodée en ajoutant un contexte formel $organisation$ (Org) – sans attributs – dont les entités correspondent aux triplets représentant les instances de la relation ternaire *organize*, ainsi que trois contextes relationnels reliant les entités de Org aux composantes initiales de *organize* : *is_done_by*$(Org, researcher)$, *is_about*$(Org, conf)$ et *with_role*$(Org, role)$. De façon similaire, la relation *publish*$(researcher, paper, inst)$ est représentée par trois contextes relationnels : *concern*$(Publish, paper)$, *is_published_by*$(Publish, researcher)$ et *with_affiliation*$(Publish, inst)$, où les entités de $Publish$ sont des triplets représentant les instances de la relation ternaire *publish*.

Avec la réification, RCA (sans intégration des relations inverses dans les données) produit 61 concepts (hors concepts bottom[2]) et inclut la réponse à *query2* mais pas celle à *query1*. Après l'ajout des relations inverses[3], RCA génère 99 concepts ; parmi eux, le concept $Inst\_12$ fournit la réponse à *query1*. Certaines requêtes nécessitent en effet l'intégration des relations inverses dans les données. Par exemple, pour *query1*, l'inverse de la relation *is_affiliated_to* a été ajouté afin de relier les entités de la catégorie $institution$ à celles de $researcher$.

La figure 9.2 présente un extrait des concepts RCA liés au concept $Inst\_12$, c'est-à-dire participant à la définition de son intension. Les flèches représentent les attributs relation-

2. Dans la suite, les concepts bottom ne sont pas pris en compte dans les résultats de RCA et de GCA.
3. Pour la suite de l'analyse, des relations inverses sont ajoutées aux données de RCA.

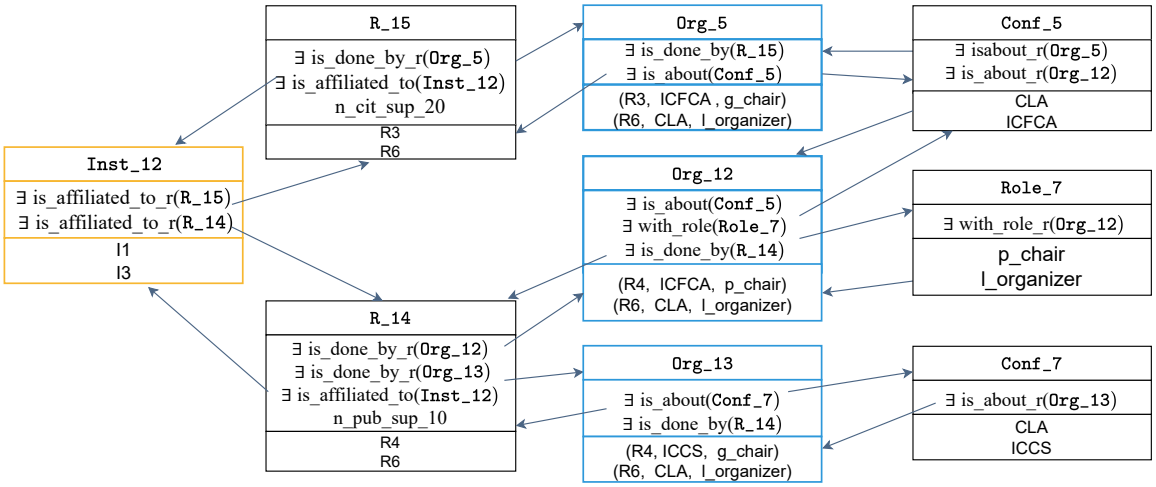

FIGURE 9.2 – Description spécifique de l'intension du concept $Inst\_12$ (*réification*).

nels entre les concepts, et les relations se terminant par "_r" correspondent aux relations inverses. L'extension de $Inst\_12$ est $\{I1, I3\}$, ce qui constitue la réponse à *query1*. Son intension indique qu'il s'agit des institutions dont les chercheurs (concepts R_14, R_15) sont reliés aux concepts de Org (concepts Org_5, Org_12, Org_13, encadrés en bleu) représentant les entités réifiées sous forme de triplets. En examinant l'intension de ces concepts Org, toutes les informations sur chaque groupe d'organisation sont accessibles : les chercheurs impliqués, les institutions concernées et les rôles des chercheurs dans l'organisation des conférences.

Dans le cas de *query2*, le résultat est $\{R5, R6\}$ qui correspond à l'extension du concept R_13, dont l'intension indique qu'il s'agit des chercheurs affiliés à des institutions françaises ($\{I1, I4\}$) et participant aux publications du concept $Publish\_8$, qui a pour extension $\{(R5, P4, I4), (R6, P4, I1)\}$. Comme pour les concepts Org, l'intension des concepts $Publish$ fournit toutes les informations relatives à une publication : les chercheurs, les articles et les affiliations associées.

Encodage par décomposition en chaîne. Les relations ternaires sont transformées en une chaîne de deux relations binaires. Ainsi, *organize*$(researcher, conf, role)$ est encodée par *is_organize_by*$(conf, researcher)$ et *has_role*$(researcher, role)$, indiquant qu'une conférence est organisée par un chercheur et qu'un chercheur possède un rôle. De même, la relation *publish*$(researcher, paper, inst)$ est encodée par *is_publish_by*$(paper, researcher)$ et *belong_to*$(researcher, inst)$, signifiant qu'un article est publié par un chercheur et qu'un chercheur appartient à une institution (comme avec la relation *is_affiliated_to*).

Avec la décomposition en chaîne, RCA produit un total de 59 concepts et inclut les réponses à *query1* et *query2*. La figure 9.3 illustre un extrait du concept R_13 dont l'extension répond à *query2*. Son intension montre que les chercheurs $\{R5, R6\}$ sont affi-

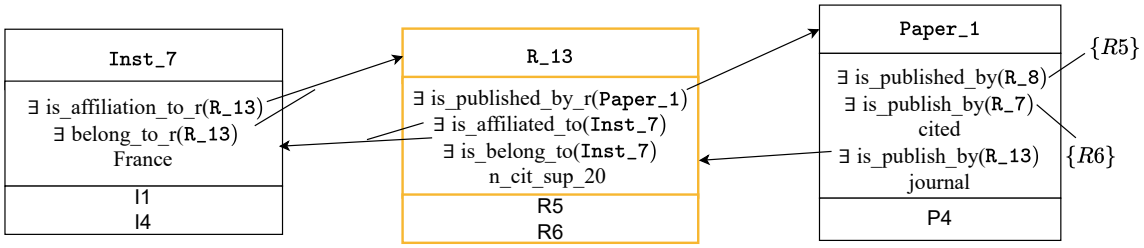

Figure 9.3 – Description spécifique de l'intension du concept R_13 (*décomposition*).

liés à des institutions françaises ($\{I1, I4\}$) et ont publié l'article P4 (concept $Paper\_1$). Contrairement à la réification, il n'y a aucune indication spécifiant qu'un article est publié par un chercheur ayant une *affiliation donnée*. En ce qui concerne *query1*, sa réponse est $\{I1, I3\}$, l'extension du concept $Inst\_11$ dont l'intension exprime qu'il s'agit des institutions ayant des chercheurs qui ont des rôles et ont organisé des conférences. Comme dans le cas de la relation *publish*, cela n'indique pas spécifiquement qu'une conférence est organisée par un chercheur ayant un *rôle donné*. Nous notons également que la relation *has_role*$(researcher, role)$ à elle seule n'est pas très significative, car un chercheur possède un rôle en ce qui concerne sa participation à l'organisation d'une conférence. En effet, la décomposition en chaîne entraîne une perte d'informations structurelles, car elle ne permet pas de capturer la dépendance globale entre les entités.

Encodage par partitionnement. Les relations sont partitionnées en fonction des entités d'une de leurs composantes. Ainsi, la relation *organize*$(researcher, conf, role)$ est scindée, selon les entités de $role$, en un ensemble de relations *role*$_i(chercheur, conf)$ pour chaque $role_i \in role$. Par exemple, *p_chair*$(R4, ICFCA)$ signifie que le chercheur $R4$ est le *program chair* de la conférence $ICFCA$. Pour la relation *publish*$(researcher, paper, inst)$, un partitionnement selon les entités de la catégorie $institution$ produit des instances de la forme *I4*$(R5, P4)$, indiquant que le chercheur $R5$ a publié l'article $P4$ avec l'affiliation $I4$. Cependant, ce partitionnement génère une forme de conflit, dans la mesure où les entités de la catégorie $institution$ interviennent également dans la relation *is_affiliated_to* : la même entité se retrouve ainsi partagée entre un objet et une relation. Pour des raisons similaires, un partitionnement de *publish* sur les entités de $researcher$ n'est pas envisageable, celles-ci étant déjà impliquées dans d'autres relations (*is_affiliated_to* et *organize*)[4]. La seule option restante consiste donc à partitionner sur les entités $paper$, ce qui conduit à des relations de la forme *paper*$_i(researcher, inst)$ pour chaque $paper_i \in paper$.

Avec ce partitionnement, RCA produit 37 concepts et contient la réponse à *query2* mais pas celle à *query1*. En effet, les entités de la catégorie $institution$ sont décrites par leurs chercheurs (via la relation *is_affiliated_to_r*), lesquels sont caractérisés par leurs *affilia-*

4. Remarquons que dans [Keip et al., 2020], le partitionnement a été appliqué à un jeu de données ne comportant qu'une seule relation ternaire, ce qui excluait toute possibilité de conflit.

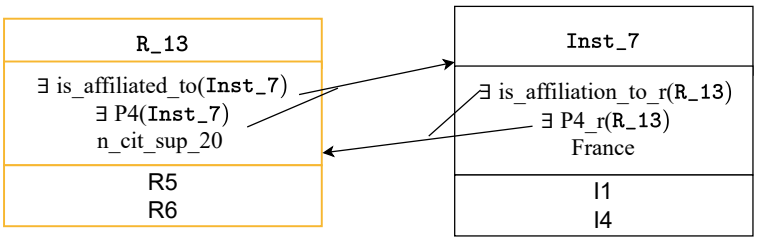

FIGURE 9.4 – Description spécifique de l'intension du concept R_13 (*partitionnement*).

tions et leurs *articles*. Cela justifie la présence d'une réponse à *query2*, portant sur les chercheurs qui ont publié des articles et sont affiliés à une institution française. En revanche, comme les chercheurs ont occupé des *rôles différents* dans l'organisation des conférences, l'ensemble des chercheurs impliqués dans l'organisation ne constitue pas un concept. Par conséquent, l'ensemble des institutions caractérisées par ces chercheurs ne forme pas non plus un concept, ce qui explique l'absence de réponse à *query1*.

En ce qui concerne *query2*, son résultat est $\{R5, R6\}$, l'extension de R_13 représentée dans la figure 9.4. Son intension indique que les chercheurs $\{R5, R6\}$ sont affiliés à des institutions françaises ($\{I1, I4\}$) et ont publié l'article $P4$ avec les affiliations $\{I1, I4\}$. Cependant, nous ne disposons d'aucune information sur les caractéristiques de $P4$, contrairement à la réification et à la décomposition en chaîne (voir figure 9.3). Le partitionnement permet de capturer la *dépendance globale entre les entités*, mais entraîne également une perte des *caractéristiques* associées aux objets des catégories partitionnées, et donc la perte des concepts correspondants. Dans notre cas, nous perdons les concepts des catégories $role$ et $paper$.

9.1.2.2 Modélisation des relations ternaires avec GCA

GCA appliqué directement sur les données initiales (*GCA-ternaire*) génère 9 graph patterns comprenant au total 63 concepts. Les réponses aux requêtes *query1* et *query2* se trouvent respectivement dans les patterns P7 (figure 9.5) et P9 (figure 9.6). À l'instar du contexte graphe, les relations ternaires apparaissent dans les graph patterns sous la forme de nœuds en ellipse.

La solution à *query1*, qui concerne les institutions dont les chercheurs ont *organisé* une conférence est $\{I1, I3\}$, correspondant à l'extension du concept $Q7a$ dans P7 (figure 9.5). Le pattern P7 capture l'organisation des conférences par des chercheurs, en indiquant les rôles et affiliations associés. Il met en évidence : (i) l'ensemble des chercheurs impliqués dans l'organisation d'au moins une conférence ($Q7m$), (ii) leurs institutions d'affiliation ($Q7a$), (iii) les conférences effectivement organisées ($Q7i$), ainsi que (iv) les rôles occupés par les chercheurs lors de ces organisations ($Q7h$).

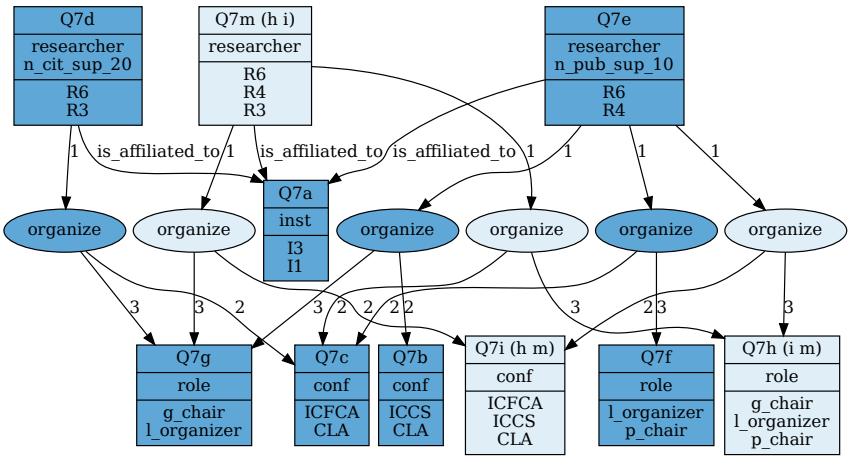

Figure 9.5 – Pattern P7 extrait des patterns du contexte graphe de la figure 9.1 .

La réponse à la question *query2*, relative aux chercheurs ayant publié des articles et affiliés à une institution française est $\{R5,\ R6\}$. Elle correspond à l'extension du concept $Q9k$ du pattern P9, représenté à la figure 9.6. Dans ce pattern, le concept $Q9k$ constitue une généralisation des concepts $Q9f$ et $Q9e$. Plus précisément, $Q9f$ regroupe les chercheurs ayant publié un article et étant affiliés à une institution française caractérisée par l'attribut *university*, tandis que $Q9e$ regroupe ceux ayant publié un article et étant affiliés à une institution française caractérisée par l'attribut *company*. Les concepts $Q9f$ et $Q9e$ permettent ainsi de répondre à des questions plus spécifiques concernant la publication d'articles par les chercheurs. Grâce à un pattern, de nombreuses questions peuvent être traitées, chaque concept représentant à la fois une question et sa réponse : la question est définie par l'intension du concept et la réponse par son extension. Par exemple, la réponse à la requête "quels chercheurs ont *organisé* des conférences et *publié* des articles" est $\{R6\}$, correspondant à l'extension de $Q9f$ (figure 9.6). Les graph patterns mettent ainsi en évidence les multiples connexions existant entre les concepts.

Comme indiqué au chapitre 4, dans la représentation hiérarchique des patterns GCA, les relations entre les concepts sont exprimées sous forme d'étiquettes textuelles dans les descriptions des concepts. Par exemple, la relation ternaire *publish*$(Q9k,\ Q9c,\ Q9i)$ reliant les concepts $Q9k$, $Q9c$, $Q9i$ dans le pattern P9 sera représentée par l'étiquette *[publish_c i]* dans l'intension de $Q9k$, ce qui signifie que : pour chaque chercheur k $\in ext(Q9k)$, il existe un article c $\in ext(Q9c)$ et une institution i $\in ext(Q9i)$ tels que la relation *publish*$(k,\ c,\ i)$ est vérifiée. Naturellement, plus l'arité d'une relation est élevée, plus il devient difficile de lire et d'interpréter les intensions des concepts en langage naturel. Cela soulève la question des encodages possibles des relations n-aires pour rendre les patterns de GCA plus lisibles. Faut-il alors représenter les relations n-aires par des relations binaires (ou d'arité inférieure

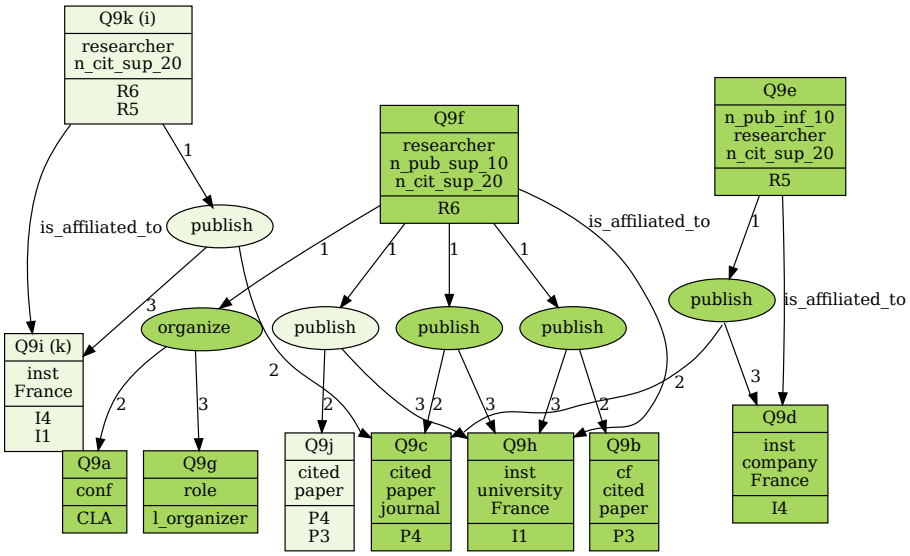

FIGURE 9.6 – Pattern P9 extrait des patterns du contexte graphe de la figure 9.1.

à n) afin de faciliter l'interprétation des patterns GCA ? Dans la comparaison suivante, nous analysons également les résultats de GCA obtenus avec les trois encodages appliqués aux données de RCA.

9.1.3 Comparaison des résultats

Pour rappel, *GCA-ternaire* (c'est-à-dire appliquée aux données initiales) produit un total de 63 concepts répartis en 9 graph patterns contenant 7 concepts automorphes (différentes occurrences d'un même concept).

Tout comme RCA, GCA produit 99 concepts à partir des données encodées par réification. Les 36 concepts supplémentaires (par rapport aux 63 concepts obtenus avec *GCA-ternaire*) proviennent des entités réifiées et n'affectent pas les concepts des catégories initiales, qui restent inchangés. De plus, le nombre de graph patterns reste identique (9), mais leur taille – c'est-à-dire le nombre de concepts par pattern – augmente avec l'ajout de nouveaux concepts, entraînant une augmentation du nombre de concepts automorphes : 110 contre seulement 7 avec *GCA-ternaire*. Ces observations mettent en évidence les limites de la réification : elle augmente la taille des données, ce qui entraîne une croissance du nombre de concepts et de la taille des graph patterns. GCA se montre peu adaptée à la réification, car en plus de l'introduction de nouveaux concepts supplémentaires, l'augmentation significative du nombre de concepts automorphes complique la lecture et l'interprétation des patterns. En revanche, la réification présente l'avantage de préserver l'intégralité des dimensions des relations sans perte d'information et de permettre la gestion de relations d'arités variées en normalisant leur représentation.

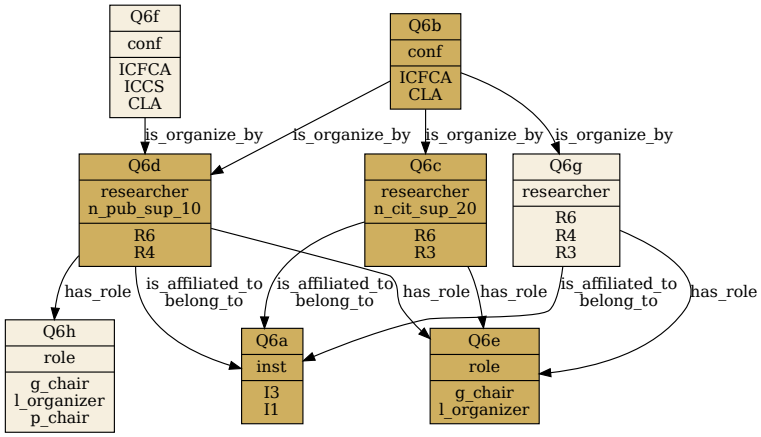

Figure 9.7 – Équivalent du pattern P7 de la figure 9.5.

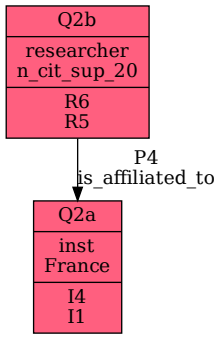

Figure 9.8 – Pattern P2.

Avec la décomposition en chaîne, GCA produit 59 concepts (identiquement à RCA) repartis en 9 patterns contenant 4 concepts automorphes. Les patterns GCA ne comportent plus de relations ternaires (nœuds en ellipse) et apparaissent ainsi plus lisibles, donc plus faciles à interpréter. La figure 9.7 présente l'équivalent du pattern P7 (figure 9.5) obtenu à partir de la décomposition en chaîne. Comme pour RCA, cette transformation entraîne une perte d'informations structurelles et peut introduire des ambiguïtés. À titre d'exemple, lorsqu'une même entité est partagée entre deux relations binaires, il devient difficile, voire impossible, de reconstituer la relation d'origine.

Pour le cas du partitionnement, GCA produit 37 concepts, 11 graph patterns, sans aucun concept automorphe. Le nombre réduit de concepts s'explique par la perte de ceux associés aux catégories partitionnées. Il en résulte des patterns plus petits et moins riches en liens entre les entités. La figure 9.8 illustre le pattern P2, qui contient la réponse à la question *query2*, représentée par l'extension du concept $Q2b$. Comme on peut le constater, les caractéristiques de l'article $P4$ ont disparu. Ce pattern P2 correspond à l'extrait RCA présenté dans la figure 9.4.

Il convient de souligner que ces différents encodages peuvent être étendus à des relations d'arité supérieure à 3, et que le choix de l'encodage dépend des questions d'analyse. Selon les requêtes à traiter, un encodage peut se révéler plus approprié qu'un autre. Par exemple, le *partitionnement* effectué sur la relation *organize* n'a pas permis de répondre à *query1*. L'analyste peut également envisager des encodages combinés. À titre d'illustration, la modélisation d'une relation $r(A, B, C, D, E)$ d'arité 5 peut commencer par sa décomposition en une chaîne de 2 relations ternaires $r1(A, B, C)$ et $r2(C, D, E)$, puis se poursuivre par la réification de $r1$ et le partitionnement de $r2$. Enfin, la différence entre le nombre de concepts obtenus avec *GCA-ternaire* (63 concepts) et ceux produits par GCA sur la décom-

position en chaîne (59 concepts) ou le partitionnement (37 concepts) illustre clairement la perte d'informations associée à ces encodages, et donc la réduction de la capacité à répondre à certaines requêtes selon la modélisation choisie. Cependant, ces encodages peuvent être utilisés dans GCA (avec des conséquences similaires à celles observées pour RCA) pour *simplifier* les graph patterns, qui deviennent plus difficiles à analyser pour les relations de *grande arité*. Il s'agit ainsi d'un compromis entre la lisibilité des patterns et la conservation des informations, comme le montre la comparaison entre le pattern de la figure 9.7 et le pattern P7 de la figure 9.5.

9.2 Traitement des cycles par RCA et GCA

L'étude des cycles et de leur longueur est essentielle dans de nombreux domaines, car ils reflètent des interactions complexes qui ne peuvent se réduire à de simples relations directes. Par exemple, dans les réseaux sociaux, les cycles révèlent des communautés fermées et des relations de réciprocité, offrant une meilleure compréhension de la structure et la cohésion des groupes. À titre d'illustration, l'équilibre des réseaux sociaux a été évalué à partir de leurs cycles simples dans [Giscard et al., 2017]. En biologie, les réseaux métaboliques présentent souvent des structures cycliques, rendant leur détection et leur analyse cruciales. Klamt et al. calculent ainsi des chemins et des cycles dans les graphes d'interactions biologiques [Klamt et von Kamp, 2009], tandis que Sridharan et al. identifient des cycles de substrats dans des réseaux métaboliques [Sridharan et al., 2015].

La comparaison des extensions de concepts RCA et GCA a montré que, même avec l'intégration des relations inverses qui rapprochent les résultats de RCA de ceux de GCA, il existe des situations où certains concepts GCA n'ont pas de correspondants dans RCA. Les analyses indiquent que ces concepts GCA absents dans les résultats de RCA présentent tous des cycles dans leurs intensions. Il est donc intéressant d'examiner plus en détail la manière dont ces deux approches traitent les cycles. Dans cette section, nous étudions, au moyen de 2 exemples, les différences de traitement des cycles par RCA et GCA, ainsi que l'effet de ces cycles sur leurs résultats. Cette analyse fait partie des travaux présentés dans [Fokou et al., 2024a].

Cycles de longueur 2-4. Le premier exemple, illustré par le contexte graphe CG_4 de la figure 9.9, comporte un cycle de longueur 2 et un cycle longueur 4, définis par la relation sociale *love*($Person, Person$). Le tableau 9.1 présente la Famille Relationnelle de Contextes (FRC_4) correspondant à CG_4.

En termes de résultats, la figure 9.11 présente l'ensemble des graph patterns GCA obtenus sur CG_4 – Q1 (bleu), Q2 (rouge), Q3 (vert citron), Q4 (vert) – tandis que la figure 9.13 montre la vue hiérarchique correspondante. Comme l'illustrent ces résultats, les patterns Q2, Q3 et Q4 contiennent des concepts automorphes, leurs cycles étant capturés via ces concepts. Par exemple, les nœuds a et b dans Q2 sont deux concepts automorphes : ils pos-

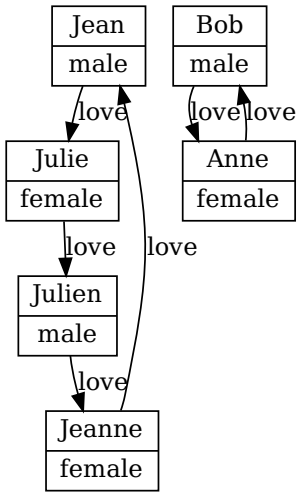

Figure 9.9 – Contexte graphe constitué des cycles de longueur 2-4 (CG_4).

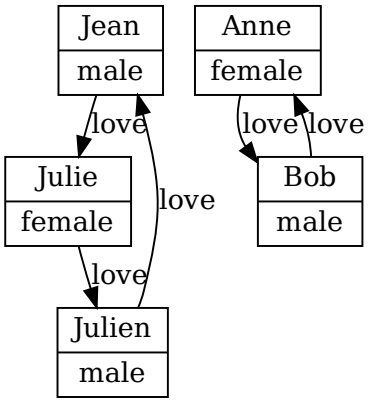

Figure 9.10 – Contexte graphe constitué des cycles de longueur 2-3 (CG_5).

Tableau 9.1 – FRC_4 (**K** = $\{Person\}$, **R** = $\{love\}$) correspondant au contexte graphe CG_4

$Person$	male	female
Jean	×	
Julie		×
Julien	×	
Jeanne		×
Bob	×	
Anne		×

love	Jean	Julie	Julien	Jeanne	Bob	Anne
Jean		×				
Julie			×			
Julien				×		
Jeanne	×					
Bob						×
Anne					×	

sèdent la même extension et des intensions équivalentes. Le pattern Q2 décrit ainsi deux personnes qui s'aiment, formant un cycle de longueur 2.

Comme décrit au chapitre 4, la duplication des concepts dans GCA est souvent nécessaire pour représenter correctement les graph patterns qui présentent des symétries. Ces concepts automorphes sont regroupés dans la vue hiérarchique sous forme de méta-nœuds (indiqués par des boîtes en pointillés), comme par exemple $Q2a$ et $Q2b$ dans la figure 9.13. Le *top concept* $Q3a$-d dans la figure 9.13, dont l'extension inclut toutes les personnes, est dupliqué en quatre nœuds formant un cycle de longueur 4 (voir la figure 9.11). Ce pattern particulier mérite une explication : en examinant les données (figure 9.9), il apparaît que toutes les personnes sont impliquées dans un cycle de longueur 2 ou de longueur 4. La généralisation la plus spécifique correspond donc à un cycle de longueur 4, puisqu'un cycle

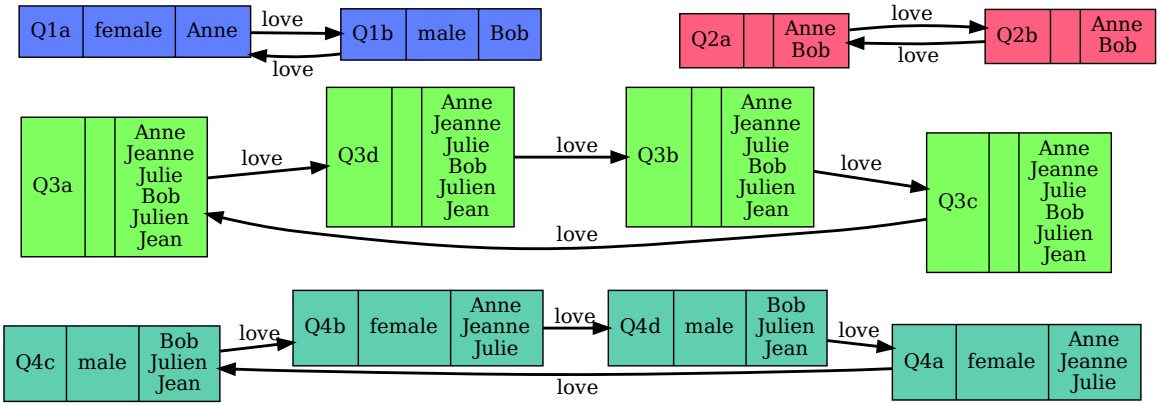

FIGURE 9.11 – Graph patterns du contexte graphe CG_4 (figure 9.9).

de longueur 2 peut simuler un cycle de longueur 4 en parcourant le cycle deux fois.

La figure 9.12 présente le treillis RCA correspondant (ici, la famille de treillis ne comprend qu'un seul treillis). Le concept $person\_3$ (top concept) possède pour attribut relationnel $\exists$ *love*($person\_3$). Il s'agit d'un concept auto-référencé, matérialisant le fait qu'une personne aime une autre personne. Le concept $person\_3$ est équivalent aux concepts automorphes qui composent le pattern Q3, car ils ont tous la même extension. Néanmoins, comme mentionné précédemment, leurs intensions sont définies différemment et véhiculent donc des informations distinctes : l'intension de $Q3a$-d intègre la taille du cycle à travers les concepts automorphes, tandis que l'intension de $person\_3$ fournit une information plus abstraite. Les concepts $person\_1$ et $person\_2$ qui se réfèrent l'un à l'autre par le biais d'attributs relationnels, représentent respectivement le fait qu'une personne *female* aime une personne *male* et vice versa. Ces concepts $person\_1$ et $person\_2$ ont respectivement les mêmes extensions que $Q4a$-b et $Q4c$-d, mais leurs intensions diffèrent. Comme pour Q3, Q4 forme un cycle de longueur 4, composé de deux paires de concepts automorphes, comme le montre la figure 9.13.

Enfin, compte tenu du fait que les concepts automorphes sont considérés comme identiques, la hiérarchie de concepts de la figure 9.13 contient **6** concepts pour GCA ($Q1a, Q1b$, $Q2a$-b, $Q4a$-b, $Q4c$-d, $Q3a$-d), contre **3** concepts pour RCA ($person\_1, person\_2, person\_3$), comme illustré dans la figure 9.12. Les concepts $Q1a, Q1b$, $Q2a$-b n'ont pas d'équivalents dans RCA, car RCA ne tient pas compte de la longueur des cycles. Par exemple, le pattern Q1 décrit qu'une femme et un homme s'aiment (cycle de longueur 2). Dans RCA, les instances $Anne$ et Bob possèdent respectivement les mêmes attributs relationnels que $Julie$, et $Julien$, à savoir $\{\exists$ *love*(*male*), $\exists$ *love_r*(*male*)$\}$, et $\{\exists$ *love*(*female*), $\exists$ *love_r*(*female*)$\}$. Par conséquent, toutes ces instances sont regroupées dans les deux concepts $person\_1$ et $person\_2$.

Les figures 9.14 et 9.15 montrent respectivement les EAR-patterns GCA et RCA obtenus

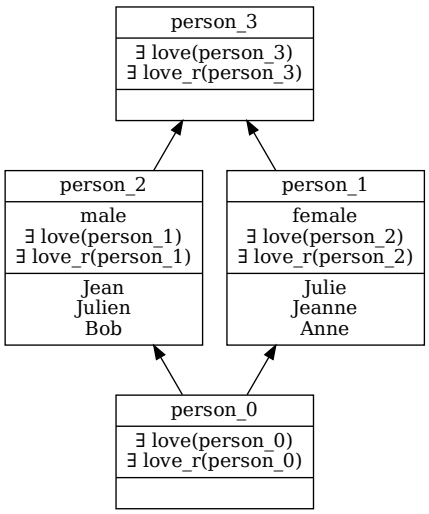

Figure 9.12 – Treillis RCA obtenu sur FRC_4 (tableau 9.1).

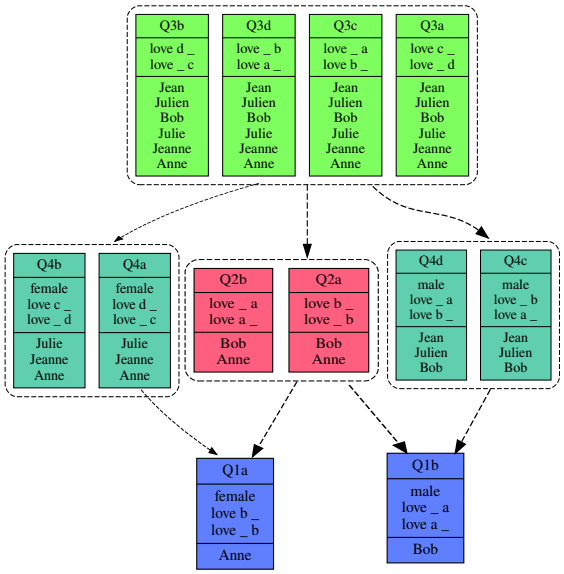

Figure 9.13 – Hiérarchie de concepts GCA obtenue sur CG_4.

pour cet exemple. Ces représentations permettent d'observer plus clairement les différences entre les deux approches. La fusion des concepts automorphes dans les EAR-patterns GCA, transforme certains cycles en boucles, généralisant ainsi la structure comme dans le résultat de RCA. C'est le cas de l'EAR-pattern GCA basé sur le concept $Q3a$ (en vert citron), qui est équivalent à l'EAR-pattern RCA construit sur le concept $person\_1$, contrairement au graph pattern initial (figure 9.11) représentant un cycle de longueur 4. En résumé, GCA produit deux EAR-patterns supplémentaires (Q1 et Q2) absents du résultat de RCA.

Cycles de longueur 2-3. Dans ce second exemple, nous modifions CG_4 afin d'étudier l'effet des cycles dont les longueurs n'ont pas de relation multiple/diviseur. Cette modification consiste à supprimer le nœud $Jeanne$, ce qui raccourcit le cycle de longueur 4 à une longueur de 3. Ce nouveau contexte graphe, nommé CG_5, est présenté à la figure 9.10, et la Famille Relationnelle de Contextes correspondante est désignée par FRC_5. Les graph patterns générés par GCA sur CG_5 sont illustrés à la figure 9.16, avec des longueurs variant entre 2, 3 et 6. Les patterns de longueur 2 et 3 capturent et généralisent la structure des cycles présents dans les données, tandis que ceux de longueur 6 généralisent les cycles de longueur 2 et 3. Par exemple, Q1 et Q4 correspondent au contexte graphe CG_5 (figure 9.10), Q2 généralise Q1, Q3 généralise Q4, Q5 généralise Q3 et Q4, et ainsi de suite. Cela montre que, dans les patterns GCA, la généralisation la plus spécifique correspond à un cycle dont la longueur est le plus petit commun multiple des longueurs des cycles dans le contexte graphe, tout comme dans CG_4 où la généralisation la plus spécifique était un cycle de longueur 4, soit le plus petit commun multiple de 2 et 4.

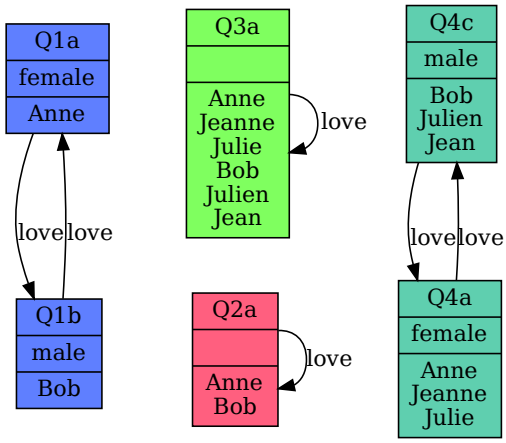

FIGURE 9.14 – Représentation des EAR-patterns associés aux graph patterns de la figure 9.11.

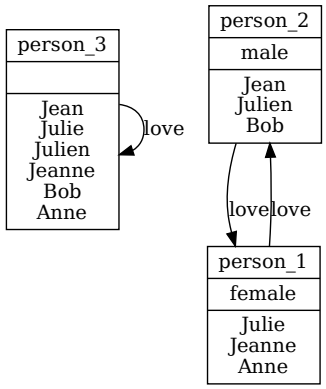

FIGURE 9.15 – EAR-patterns associés au treillis RCA de la figure 9.12.

Contrairement à l'exemple CG_4, où le concept d'extension $\{Anne\}$ ($Q1a$ dans la figure 9.13) n'avait pas d'équivalent dans les résultats de RCA (figure 9.12), il est bien présent dans les résultats RCA obtenus sur FRC_5 (concept $person\_11$ dans la figure 9.17). En effet, lors du processus itératif, les attributs relationnels de $\{Anne\}$ et $\{Bob\}$ diffèrent de ceux des autres instances, car ils pointent vers des concepts distincts. RCA capture ainsi la différence entre les deux cycles : *une femme aime un homme qui aime une femme* versus *une femme aime un homme qui aime un homme* , sans tenir compte de leur longueur. GCA produit deux concepts supplémentaires par rapport à RCA sur CG_5 (23 concepts contre 21) : les concepts automorphes $Q2a$-b et $Q5a$-c (figure 9.16) représentent respectivement les personnes impliquées dans des cycles de longueur 2 et 3, indépendamment de leur genre. Dans RCA, ces deux patterns sont généralisés dans le top concept auto-référencé, qui peut être interprété comme une *personne qui aime une personne et qui est aimée par une personne.*

Ces analyses permettent de conclure que, lorsque les données contiennent des cycles, certains graph patterns GCA peuvent représenter des cycles que RCA ne peut pas exprimer explicitement, puisque les cycles en RCA sont capturés via des attributs relationnels renvoyant à des concepts. Là où GCA utilise des concepts automorphes pour représenter distinctement certains cycles, RCA les généralise en des boucles simples ou les perd. C'est le cas des patterns Q2 et Q5 (figure 9.16), qui ne sont pas explicitement représentés dans le résultat de RCA, mais sont englobés dans le top concept. Ainsi, GCA conserve des informations sur la longueur des cycles, ce qui n'est pas le cas dans RCA.

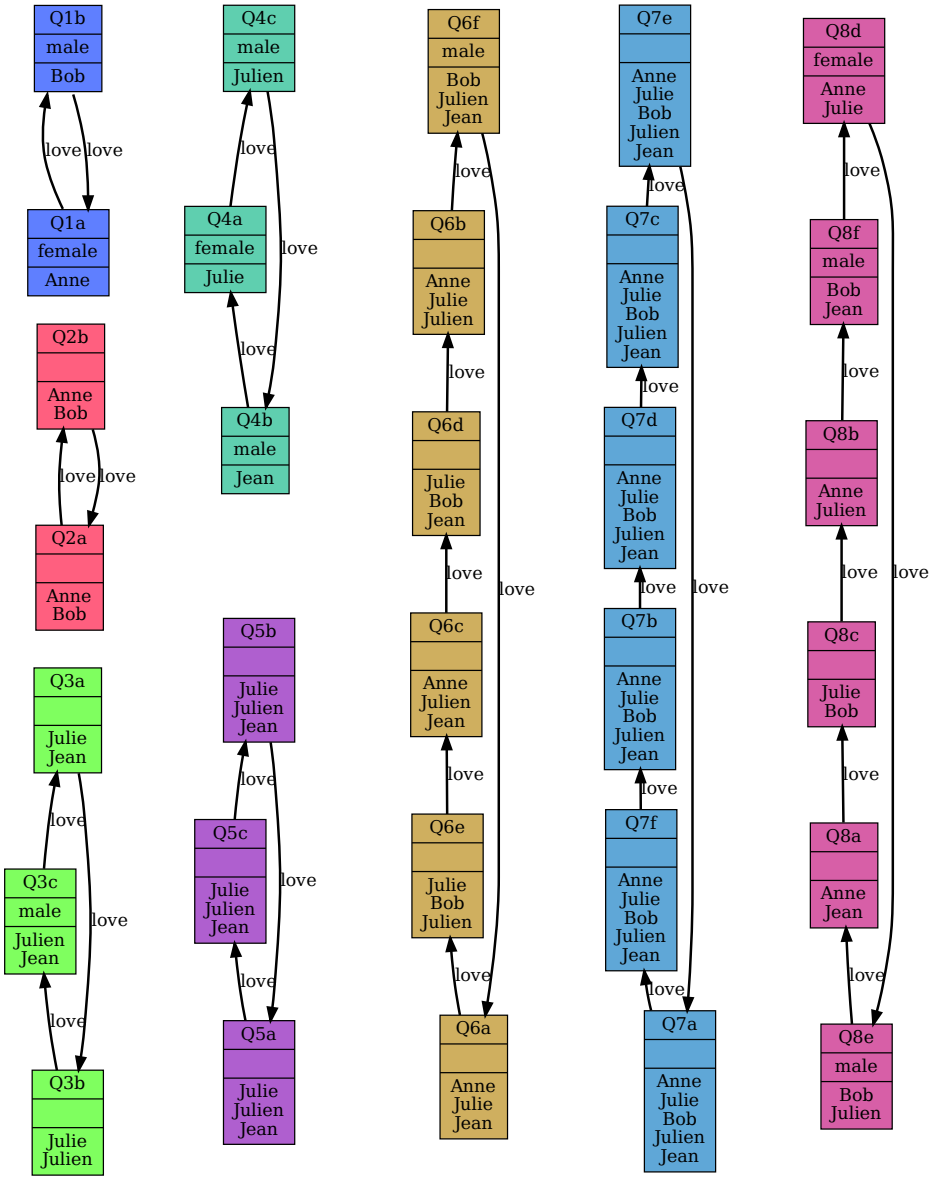

Figure 9.16 – GCA patterns for CG_5.

9.3 Mise en œuvre de RCA et GCA sur un jeu de données réel

Dans cette section, nous mettons en œuvre RCA et GCA sur un jeu de données réel, issu d'une ancienne pharmacopée arabe [Kahl, 2009]. Ces données ont déjà été explorées à l'aide de l'Analyse Formelle de Concepts (AFC) et de RCA dans [Fokou et al., 2024b], dans le but d'extraire les connaissances reliant les symptômes aux ingrédients des remèdes qui les

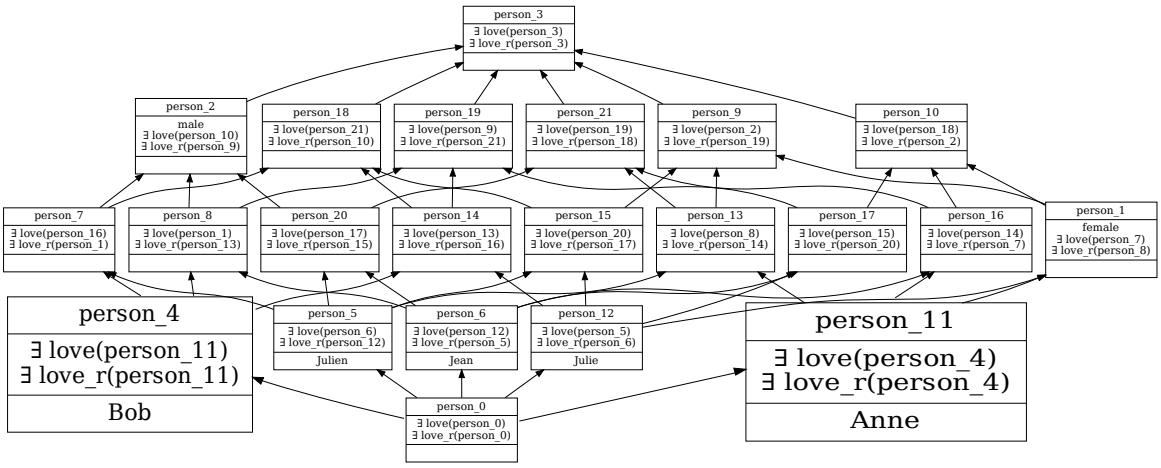

FIGURE 9.17 – Treillis RCA obtenu sur FRC_5 correspondant à CG_5 où les concepts 4 et 11 sont mis en évidence.

traitent. L'objectif était de répondre à des questions formulés par des biologistes, telles que : "existe-t-il des groupes d'ingrédients pouvant être associés à des groupes de symptômes ?". Nous présentons ici la synthèse des résultats obtenus avec RCA, ainsi que les difficultés rencontrées lors de l'application de GCA sur ces mêmes données.

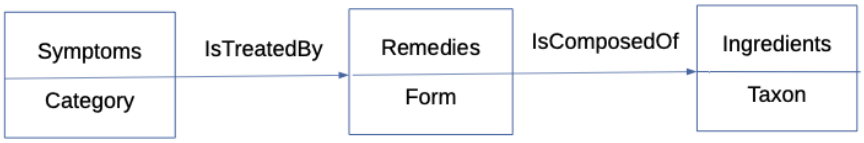

FIGURE 9.18 – Modèle de données de FRC_7 reliant les symptômes aux ingrédients.

Le modèle de données utilisé encode les relations entre les symptômes, les remèdes et les ingrédients comme l'illustre le diagramme de la figure 9.18. Pour cette analyse, les données ont été limitées à un sous-ensemble de remèdes traitant les symptômes de la *fièvre*, leurs ingrédients et les symptômes associés. La FRC qui en découle est définie par $(\mathbf{K}, \mathbf{R}) = (\{\mathcal{K}_{Symptoms}, \mathcal{K}_{Remedies}, \mathcal{K}_{Ingredients}\}, \{isTreatedBy, isComposedOf\})$ et est désignée par FRC_7. Le contexte $\mathcal{K}_{Symptoms}$ (105×9) décrit les symptômes par leur type (catégorie) – ces catégories sont basées sur l'expertise – le contexte $\mathcal{K}_{Remedies}$ (26×13) décrit les remèdes par leurs formes et le contexte $\mathcal{K}_{Ingredients}$ (156×147) décrit les ingrédients par leurs taxons (espèce et famille). Concernant les contextes relationnels, le contexte *isTreatedBy* décrit la relation selon laquelle un symptôme est traité par un remède, tandis que le contexte *isComposedOf* exprime le fait qu'un remède est composé de certains ingrédients.

Les expériences ont été réalisées sur un ordinateur équipé d'un processeur Intel Core i7-12850HX (16 Cœurs avec Hyper-Threading, frequence 2.1-4.8GHz) et de 16Go de mémoire RAM DDR5.

9.3.1 Analyse avec RCA

Pour l'analyse avec RCA, deux combinaisons de quantificateurs de scaling ont été utilisées afin d'extraire différents niveaux d'information : (1) le quantificateur $\exists$ est appliqué aux deux relations, (2) le quantificateur $\exists\forall$ est appliqué à la relation *isTreatedBy*, tandis que le quantificateur $\exists$ est utilisé pour *isComposedOf*.

Quantificateur $\exists$. En termes de résultats, la famille de treillis de concepts obtenue contient des séquences d'informations suivant la structure du diagramme illustré à la figure 9.18, les relations étant quantifiées de manière existentielle. Cette configuration permet d'extraire des connaissances de la forme : pour chaque $x \in Ext(Symptoms\_i)$, il existe un *remède* y qui traite x et il existe un *ingrédient* z qui compose y, reliant ainsi les symptômes et les ingrédients à travers les remèdes. RCA produit un total de 1126 concepts, repartis comme suit :

- 837 concepts pour le contexte $\mathcal{K}_{Symptoms}$;
- 168 concepts pour le contexte $\mathcal{K}_{Remedies}$;
- 121 concepts pour le contexte $\mathcal{K}_{Ingredients}$.

Afin de faciliter l'analyse, un treillis Iceberg (avec un seuil de 4 %) [Stumme et al., 2002] a été construit sur le contexte $\mathcal{K}_{Symptoms}$. La figure 9.19 illustre un extrait de concepts connectés issus des résultats de RCA. Elle met en évidence les symptômes (extension du concept $Symptoms\_39$) traités par au moins un remède appartenant au concept $Remedies\_105$ (ainsi qu'à ses sur-concepts $Remedies\_130$ et $Remedies\_165$), lesquels sont composés d'au moins un ingrédient provenant des familles *apiaceae* ($Ingredients\_121$), *zingiberaceae* ($Ingredients\_102$) et *piperaceae* ($Ingredients\_112$). Cet extrait suggère que les ingrédients issus de ces familles de plantes sont utiles pour le traitement des symptômes regroupés dans $Symptoms\_39$. En effet, le poivre (*peper*) et la cardamome (*cardamom*), par exemple, sont bien connus pour leurs effets bénéfiques sur les troubles digestifs.

Nous avons ensuite intégré **les relations inverses dans FRC$_7$** afin d'analyser les résultats et d'observer le comportement de RCA dans ce contexte. L'exécution de RCA a été bloquée après la troisième itération de scaling dans l'outil RCAExplore [Dolques et al., 2019] (jusqu'à interruption manuelle du processus). De même, dans l'outil FCA4J [Gutierrez et al., 2022], l'exécution a échoué après la troisième itération de scaling avec une erreur d'allocation mémoire. À ce stade, le nombre total de concepts générés atteint **42 445**, repartis comme suit :

- **3443** concepts contre 837 pour le treillis de $\mathcal{K}_{Symptoms}$;
- **16 720** concepts contre 168 pour le treillis de $\mathcal{K}_{Remedies}$;
- **22 282** concepts contre 121 pour le treillis de $\mathcal{K}_{Ingredients}$.

Ces statistiques montrent que le nombre de concepts est déjà considérable dès la troisième itération de scaling, ce qui illustre le caractère potentiellement explosif du calcul à l'itération suivante, tant en termes de taille des contextes formels (en nombre d'attributs relationnels) qu'en taille des treillis produits. On peut donc conclure, à la lumière de ce cas

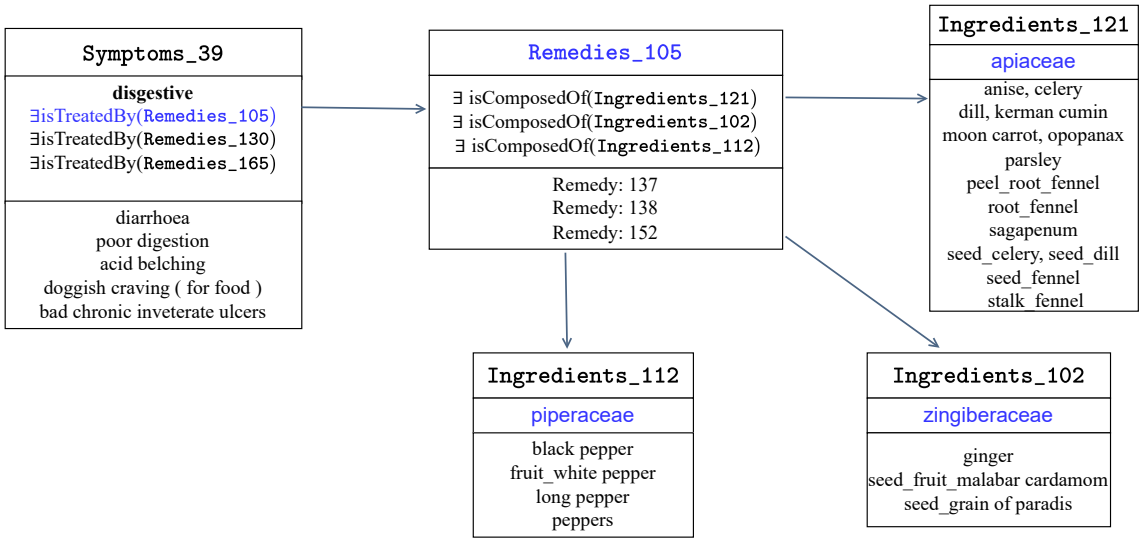

FIGURE 9.19 – Extrait des résultats de RCA sur FRC_7 avec les quantificateurs ∃ / ∃.

applicatif, que l'ajout systématique des relations inverses n'est pas toujours pertinent dans les applications pratiques de RCA, au risque pour l'analyste d'être rapidement submergé par un nombre excessif de concepts. Par ailleurs, selon les objectifs d'analyse, certaines relations inverses peuvent ne pas être significatives.

Quantificateurs ∃∀ et ∃. L'utilisation du quantificateur ∃∀ sur la relation *isTreatedBy* et du quantificateur ∃ sur la relation *isComposedOf* permet d'extraire des séquences d'informations sous la forme suivante : pour chaque $x \in Ext(Symptoms\_i)$, tous les *remèdes* y qui traitent x appartiennent à $Ext(Remedies\_j)$, et il existe au moins un *ingrédient* $z \in Ext(Ingredients\_p)$ qui compose y. Ainsi, l'ingrédient z est utile pour le traitement du symptôme x.

Le concept $Symptoms\_31$ illustré à la figure 9.20, possède une extension plus restreinte que celle du concept $Symptoms\_39$ présenté à la figure 9.19, bien que leurs attributs relationnels renvoient aux mêmes concepts du treillis de $\mathcal{K}_{Remedies}$. En effet, avec l'utilisation du quantificateur ∃∀ sur la relation *isTreatedBy*, tous les remèdes qui traitent les symptômes de $Symptoms\_31$ appartiennent à l'extension du concept $Remedies\_105$ (ainsi qu'à celles de ses sur-concepts $Remedies\_130$ et $Remedies\_165$). Cette contrainte explique l'absence du symptôme $diarrhoea$ dans l'extension de $Symptoms\_31$, car celui-ci est traité à la fois par $Remedy$ 152 et par un autre remède n'appartenant pas au concept $Remedies\_105$. Finalement, ce concept fournit une information plus précise que celle obtenue avec le quantificateur ∃. Il permet de conclure qu'il existe au plus trois remèdes qui traitent ces quatre symptômes digestifs, et que leurs ingrédients proviennent de quelques familles de plantes bien identifiées. Ce résultat peut conduire à une étude plus approfondie de ces remèdes spécifiques ainsi que des ingrédients qu'ils partagent, afin de mieux comprendre leur efficacité

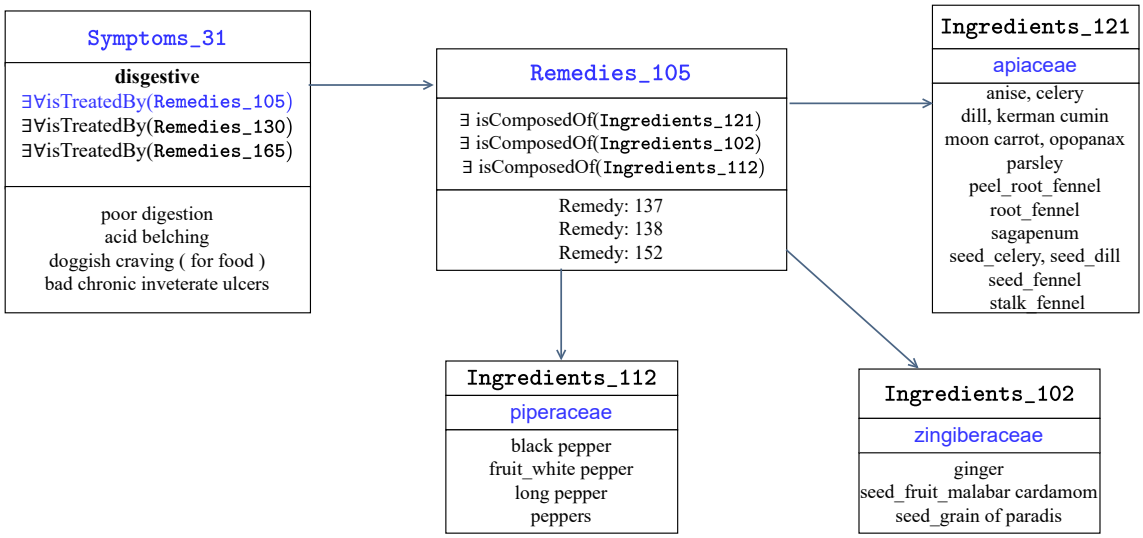

Figure 9.20 – Extrait des résultats de RCA sur FRC_7 avec les quantificateurs ∃∀ / ∃.

potentielle dans le traitement des troubles digestifs.

9.3.2 Analyse avec GCA

Pour analyser ces données avec GCA, la FRC_7 a été transformée afin d'obtenir le contexte graphe correspondant, noté CG_7. La figure 9.21 illustre un extrait partiel de ce contexte graphe. Celui-ci se compose d'une grande composante connectée, relativement dense en relations, ainsi que de quelques nœuds isolés.

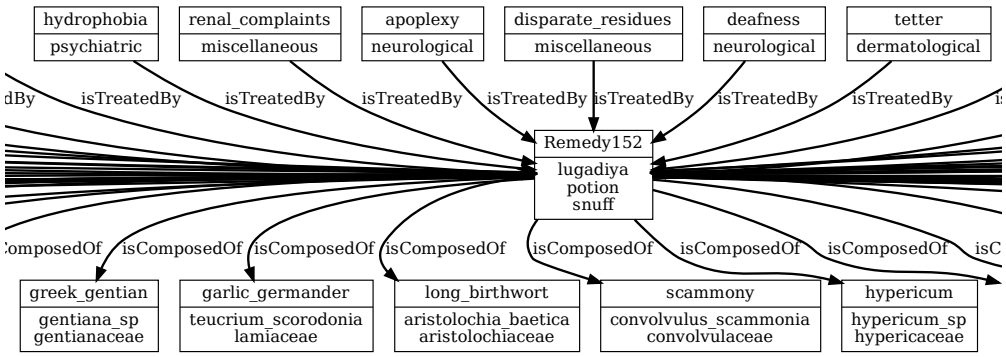

Figure 9.21 – Extrait du contexte graphe CG_7 correspondant à RFC_7.

L'exécution de GCA sur ce contexte graphe ne produit aucun résultat, même partiel. Le processus est bloqué dès les premières étapes, plus précisément lors du premier calcul du produit catégorique $I_7 \times I_7$, où I_7 désigne la relation d'incidence de CG_7. Comme GCA vise

à capturer les structures présentes dans le contexte graphe, le pattern résultant du produit représenterait donc une structure de très grande taille, correspondant à la forme globale de CG_7 (voir figure 9.21). Or, CG_7 contient 287 sommets, ce qui implique que le produit $I_7 \times I_7$ génère un graphe de 287^2 sommets, rendant le calcul extrêmement coûteux en mémoire et en temps de traitement. De plus, pour calculer le concept unaire correspondant à un nœud x d'un pattern P, il est nécessaire de déterminer le plus petit retract (définition 4.3) de P contenant x, qui sert à définir l'intension du concept. Cette opération accroît encore la complexité, en raison de la taille du graphe et du nombre élevé de retracts à calculer.

Comme nous l'avons vu précédemment, l'intégration des relations inverses conduit RCA à produire **42 445** concepts dès la troisième itération de scaling, avant que son exécution ne se bloque. Cela signifie que, si l'on suppose que l'ensemble des concepts produits par RCA est équivalent à celui de GCA pour cet exemple, le résultat de GCA correspondrait alors un à graph pattern d'au moins **42 445** nœuds (concepts). Compte tenu de la structure du graphe, il est raisonnable de penser que le nombre de concepts automorphes y serait non négligeable, ce qui complexifierait encore davantage la structure du graph pattern résultant, tant en taille qu'en densité des relations.

Pour vérifier l'hypothèse concernant la présence de nombreux concepts automorphes, nous avons extrait un sous-graphe de CG_7, composé de 24 nœuds et de 66 arêtes (unaires et binaires), que nous appelons CG_8. Sur cet extrait, l'exécution de GCA parvient à calculer (en quelques secondes) une partie des résultats avant de se bloquer. L'analyse de ce résultat partiel révèle que GCA a produit un graph pattern de 93 nœuds, comprenant 63 concepts unaires et 20 concepts automorphes.

Du côté de RCA, l'analyse produit une famille de treillis comptant au total 42 concepts lorsque les relations inverses ne sont pas prises en compte. L'intégration des relations inverses fait passer ce nombre à 63. Le calcul des **EAR-patterns** sur cette famille de treillis aboutit à un unique EAR-pattern, qui met en évidence les connexions entre les 63 concepts. La figure 9.22 présente un extrait de ce EAR-pattern RCA, où les noms de relations sont abrégés en *isC* pour *isComposedOf* et en *isT* pour *isTreatedBy*, afin d'un faciliter la lecture. Dans cet extrait, on observe les informations relatives au traitement du symptôme $putrid\_fevers$, à savoir que ce symptôme est traité entre autres par les remèdes $Remedy\,16$ et $Remedy\,20$, tous deux sous forme de *pastille*, et composés des ingrédients du concept $ingr\_10$. En particulier, les ingrédients de $Remedy\,20$ proviennent de la famille *apiaceae*.

Ce résultat permet de constater que la difficulté de GCA ne dépend pas seulement de la taille du contexte graphe, mais surtout de sa structure, notamment les relations de type *n-n* (plusieurs-à-plusieurs). D'autres expériences menées dans le cadre de ce travail ont montré que GCA fonctionne moins efficacement lorsque le contexte graphe se compose d'une grande composante connectée avec de nombreuses connexions, par rapport à des graphes constitués d'ensemble de petites composantes. Cette observation a également été corroborée sur les variantes de l'exemple concernant les voitures et les garages, analysées dans la section 7.3.3.

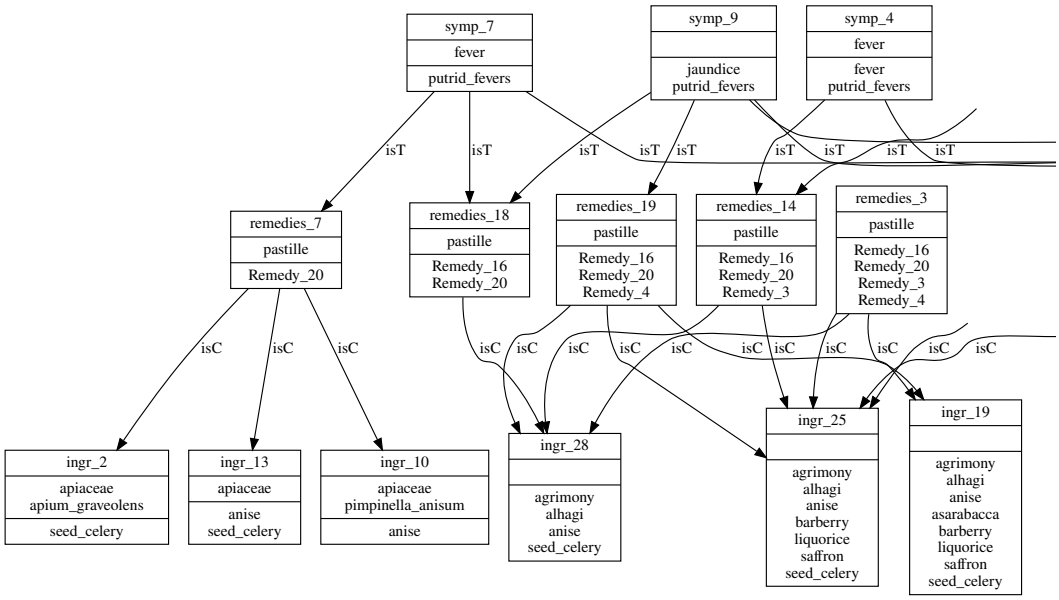

Figure 9.22 – Extrait du EAR-pattern RCA obtenu sur FRC_8 correspondant à CG_8.

Dans de tels cas "pathologiques", où GCA peine à compléter les calculs, une solution pourrait consister à calculer d'abord les familles de treillis RCA, puis à en déduire les **EAR-patterns** associées. Ces derniers mettent en évidence les structures relationnelles entre les concepts, de la même manière que les graph patterns GCA. Pour un exemple comme celui traité ici, le EAR-pattern RCA résultant serait constitué de minimum **42 445** concepts avec leurs relations correspondantes. En pratique, une telle structure est difficilement exploitable par un expert métier. Il serait donc par exemple intéressant d'envisager des solutions permettant d'effectuer des requêtes ciblées sur ces structures afin de faciliter l'analyse.

9.4 Atouts et limites de RCA et GCA d'un point de vue pratique

D'un point de vue pratique, RCA et GCA présentent chacun des avantages et des limites. Cette section en expose quelques-uns

Complexité de calcul. Le calcul du produit catégorique de graphes devient rapidement coûteux lorsque les graphes sont de grande taille et présentent de nombreuses connexions entre les éléments. C'est le cas de l'exemple CG_7 présenté précédemment, pour lequel l'exécution de GCA a été bloquée par manque de ressources. En revanche, dans de tels scénarios où le nombre de concepts est très important, RCA peut pallier ces problèmes de performances et de lisibilité en permettant de filtrer les concepts, par exemple en construisant

des treillis Iceberg [Stumme et al., 2002] ou des sous-hiérarchies de Galois [Godin et Mili, 1993].

Les quantificateurs de scaling. Contrairement à GCA, qui n'utilise implicitement que le quantificateur $\exists$, RCA bénéficie d'une diversité de quantificateurs ($\exists$, $\exists\forall$, $\exists\forall_{\geq n\%}$, etc.) [Braud et al., 2018 ; Rouane-Hacene et al., 2013]. Cette variété confère au processus d'analyse une grande flexibilité en termes de précision de l'information qui peut être extraite. Ainsi, ces quantificateurs permettent à l'analyste de capturer différents niveaux d'informations.

Lisibilité des résultats. La représentation principale des résultats de GCA sous forme de graph patterns offre une vue compacte des concepts et permet d'obtenir une vision globale des structures présentes dans les données, ce qui constitue un avantage pour l'analyse. En revanche, l'interprétation des résultats de RCA peut être plus difficile lorsque l'ensemble de données est volumineux, en raison de la nécessité de naviguer entre plusieurs treillis. C'est pourquoi des outils tels que RCAviz [Huchard et al., 2024] ont été développés pour faciliter cette navigation. Toujours dans cet objectif de rendre l'interprétation des résultats de RCA plus accessible, Gutierrez et al. se sont intéressés à la génération de descriptions en langage naturel des artefacts RCA à l'aide des *Large Language Models* (LLMs) [Gutierrez et al., 2025].

Modélisation des relations. RCA et GCA modélisent les relations de différentes manières. La capacité de GCA à traiter les relations n-aires d'arité quelconque constitue un avantage majeur pour l'analyse, car permet d'éviter les encodages préalables qui peuvent entraîner une perte d'information. Cependant, l'interprétation des résultats devient plus complexe à mesure que l'arité des relations augmente, ce qui peut nécessiter le recours à des encodages tels que le *partitionnement* ou la *décomposition* pour réduire l'arité des relations. Aussi, le fait que GCA intègre automatiquement les relations inverse peut s'avérer problématique pour des données contenant des relations symétriques, ces dernières étant alors prises en compte en double. Enfin, pour l'analyse des données comportant des cycles, RCA et GCA peuvent être utilisés de manière complémentaire afin de capturer différents niveaux de longueur des cycles.

Concept n-aires. RCA se limite au calcul de concepts unaires, tandis que GCA calcule également des concepts n-aires (ou concepts de relations n-aires), dont les extensions sont constituées d'ensembles de n-uplets d'objets. La possibilité de calculer les concepts n-aires constitue un avantage, car elle permet de représenter des structures plus riches et, par conséquent, de répondre à des requêtes d'analyse plus complexes.

9.5 Conclusion

Dans ce chapitre, nous avons réalisé une analyse comparative de RCA et de GCA en nous focalisant sur leurs différences, notamment en ce qui concerne la modélisation des

relations n-aires et le traitement des cycles.

Dans un premier temps, nous avons comparé RCA et GCA sur la modélisation des relations ternaires, en examinant leur capacité à répondre aux questions d'analyse. En effet, GCA permet de traiter directement les relations ternaires (et, de manière générale, les relations n-aires), tandis que RCA se limite aux relations binaires. Nous avons utilisé trois encodages (*réification, décomposition en chaîne* et *partitionnement*) pour modéliser les données de RCA, et nous avons analysé l'impact de ces encodages en termes de perte d'informations, d'augmentation de la taille des données et, par conséquent, de capacité à résoudre les requêtes. Nous avons ensuite appliqués ces mêmes encodages aux données de GCA. Les résultats montrent que les encodages par *décomposition* et par *partitionnement* produisent des graph patterns plus lisibles et plus faciles à interpréter.

Dans un second temps, nous avons analysé les différences de traitement des cycles entre les deux approches. Cette analyse a révélé que GCA capture certains cycles qui ne peuvent pas être représentés par RCA, dans la mesure où, en RCA, les cycles sont pris en compte par le biais d'attributs relationnels revoyant à des concepts. Elle a également montré que, là où GCA utilise des concepts automorphes pour représenter explicitement certains cycles, RCA les généralise en boucles simples. Par ailleurs, nous avons évoqué l'impact positif de la prise en charge des concepts n-aires par GCA et de l'utilisation de divers quantificateurs dans RCA sur la qualité de l'analyse. En conclusion, les différences entre RCA et GCA apparaissent comme complémentaires et peuvent être exploitées conjointement pour enrichir une tâche d'analyse.

Chapitre 10

Conclusion et Perspectives

10.1 Conclusion

L'Analyse Relationnelle de Concepts (RCA) et l'analyse conceptuelle des graphes (GCA) constituent deux extensions majeures de l'Analyse Formelle de Concepts (AFC), développées pour répondre au besoin croissant de traiter des données multi-relationnelles. Comme discuté au chapitre 5, les travaux existants cherchant à rapprocher ou comparer RCA et GCA se sont concentrés sur des aspects spécifiques, notamment l'interprétation des résultats. Cependant, aucune étude n'avait jusqu'à présent proposé une comparaison globale et approfondie, ni sur leurs fondements théoriques, ni sur la nature des résultats produits. L'étude menée dans le cadre de cette thèse a ainsi permis d'élargir la portée de cette comparaison, en offrant une vision plus complète des similarités et des différences entre ces deux approches, aussi bien du point de vue théorique que pratique.

La première partie a présenté le cadre général du travail. Le chapitre 2 a introduit les notions fondamentales de l'Analyse Formelle de Concepts (AFC), ainsi qu'un aperçu de ses principales extensions destinées aux données multi-relationnelles. Les chapitres 3 et 4 ont ensuite présenté, respectivement, les approches RCA et GCA, qui constituent le socle de ce travail. Enfin, le chapitre 5 a proposé une synthèse de l'état de l'art sur les liens et les rapprochements entre RCA et GCA.

La deuxième partie de cette thèse a présenté les principaux résultats obtenus. Ces contributions concernent à la fois les aspects théoriques et pratiques des méthodes RCA et GCA. Elles sont résumées ci-dessous.

Inclusion de l'ensemble des concepts RCA dans celui de GCA. Nous avons démontré que, pour un même jeu de données, l'ensemble des concepts produits par RCA est inclus dans celui généré par GCA sous un paramétrage commun, ce qui met en évidence le caractère expressif de GCA par rapport à RCA [Fokou et al., 2025b]. Ce paramétrage commun repose notamment sur :

- l'utilisation du quantificateur existentiel ($\exists$) ;
- le calcul des concepts unaires ;
- le traitement des relations binaires ;
- et l'intégration des relations inverses dans les données de RCA.

Cette dernière condition permet de compenser la prise en compte implicite des relations inverses dans le processus de GCA, assurant ainsi une base commune pour la comparaison des deux approches.

Transformation de la famille de treillis RCA en graphes. Dans cette base commune, nous avons proposé et formalisé une transformation de la famille de treillis de concepts produite par RCA en un ensemble de patterns relationnels, appelés EAR-patterns. Ces patterns offrent une représentation plus compacte et lisible des résultats de RCA, constituant ainsi une base solide pour améliorer leur interprétation et leur visualisation.

Modélisation des relations n-aires. L'étude menée sur la modélisation des relations n-aires dans RCA et GCA nous a permis de montrer que les encodages couramment utilisés en RCA (la *réification*, la *décomposition* et le *partitionnement*) ont un impact significatif sur la capacité à répondre aux requêtes d'analyse. Nous avons également montré que les encodages par décomposition et partitionnement, lorsqu'ils sont appliqués dans GCA, permettent de produire des graph patterns plus lisibles et plus faciles à interpréter. Cela suggère que ces stratégies d'encodage pourraient être exploitées en GCA pour faciliter l'analyse et la compréhension des résultats, car la complexité des graph patterns augmente avec l'arité des relations.

Traitement des cycles. Notre étude sur le traitement des cycles a révélé que GCA est capable de capturer certains cycles que RCA ne peut pas représenter, du fait que, dans RCA, les cycles sont modélisés à travers des attributs relationnels renvoyant à d'autres concepts. Par ailleurs, GCA utilise des concepts automorphes pour représenter explicitement certains cycles, tandis que RCA tend à les généraliser sous la forme de boucles simples. Ces observations indiquent que, pour des tâches d'analyse visant à identifier ou caractériser la longueur et la structure des cycles, GCA se révèle plus adaptée.

Dans l'ensemble, ces résultats montrent que RCA et GCA sont complémentaires, chacune présentant des avantages spécifiques selon les objectifs d'analyse. Leur utilisation conjointe pourrait ainsi permettre d'enrichir les tâches d'analyse.

10.2 Perspectives

Les résultats issus de cette étude comparative entre RCA et GCA ouvrent plusieurs pistes de recherche prometteuses. Ils invitent notamment à approfondir les questions d'interopérabilité, de complémentarité et de combinaison entre les différentes extensions de l'AFC appliquées aux données multi-relationnelles, et plus particulièrement entre RCA et GCA. Dans cette section, nous présentons quelques perspectives que nous jugeons intéressantes à explorer dans la continuité de ce travail.

Extension des EAR-patterns RCA à d'autres quantificateurs. Les EAR-patterns RCA constituent une base solide pour faciliter l'interprétation de la famille de treillis générée par RCA. Cependant, leur définition actuelle se limite au quantificateur existentiel ($\exists$). Une première perspective intéressante consisterait à étendre les EAR-patterns aux familles de treillis construites avec d'autres quantificateurs tels que $\exists\forall$, $\exists\ \supseteq$, etc. [Braud et al., 2018], afin de calculer des patterns relationnels plus représentatifs des structures présentes dans les données. Pour cela, il faudrait redéfinir la notion de redondance des attributs relationnels pour chaque quantificateur. Par ailleurs, l'ajout des relations inverses dans les données de RCA, bien qu'essentiel pour le calcul des composantes fortement connexes servant de base pour la définition des RCA-patterns, n'est pas toujours pertinent en pratique. L'étude menée sur les données issues d'une pharmacopée a montré que cet ajout peut, d'une part, accroître de manière excessive le nombre de concepts – rendant l'analyse plus complexe – et, d'autre part, ne pas toujours avoir de sens selon les objectifs d'analyse. Ainsi, une alternative consisterait à intégrer les arêtes inverses au niveau du graphe de dépendances, plutôt que directement dans les données, sous certaines conditions de cohérence. Une telle approche permettrait d'obtenir des EAR-patterns plus petits, fidèles aux données initiales (sans relations inverses) et donc plus faciles à interpréter.

Modélisation. Il serait intéressant d'étendre et d'approfondir, sur des jeux de données réels, l'étude comparative de RCA et GCA en se focalisant sur l'impact de la modélisation des relations ternaires sur leur capacité à répondre aux questions d'analyse. Cette investigation permettrait d'évaluer à la fois la complexité induite par les différents encodages et la lisibilité réelle des résultats. En parallèle, il serait intéressant d'étudier l'impact d'autres différences clés entre RCA et GCA, telles que l'utilisation des quantificateurs, le calcul des concepts n-aires, ou encore les différents paramètres proposés par les outils qui implémentent ces deux approches. L'objectif serait de définir une catégorisation des contextes applicatifs, afin d'identifier pour chaque type de données ou question d'analyse l'approche la plus adaptée, RCA ou GCA. D'un point de vue pratique, le développement d'un cadre complet d'analyse permettant de choisir, de manière guidée, les encodages (*réification*, *décomposition*, *partitionnement*, etc.), les méthodes (RCA, GCA, etc.), les quantificateurs de scaling et d'autres paramètres en fonction des questions d'analyse serait très utile pour les experts métier. Un tel cadre devrait notamment offrir la possibilité à l'analyste de simuler différents scénarios d'analyse afin de sélectionner celui qui répond le mieux à ses besoins.

Concepts n-aires. La capacité à calculer les concepts n-aires constitue un atout pour l'analyse, car ces concepts permettent de représenter des structures plus riches et de répondre à des requêtes d'analyse plus complexes. Actuellement, RCA se limite aux concepts unaires, tandis que GCA définit des concepts n-aires. Une perspective intéressante serait d'étendre cette notion à RCA afin de combiner l'avantage des concepts n-aires de GCA avec la flexibilité offerte par les quantificateurs de RCA. Formellement, ces concepts n-aires pourraient être définis en s'appuyant sur les EAR-patterns RCA et en utilisant un calcul de projection de graphes similaire à celui employé dans GCA. D'un point de vue pratique, pour que

l'analyste puisse exploiter pleinement ces concepts n-aires, il serait nécessaire de disposer d'une plateforme de navigation des graphes comparable à RCAviz [Huchard et al., 2024]. Une telle plateforme permettrait d'explorer le voisinage d'un ou plusieurs concepts au sein des patterns, et de calculer un concept n-aire en sélectionnant les propriétés correspondant à son intension (nœuds et arêtes). Concrètement, cela reviendrait à construire et évaluer une requête conjonctive ciblée, limitant ainsi le calcul aux concepts n-aires pertinents par rapport aux objectifs d'analyse de l'utilisateur.

Version déclarative de RCA. RCA est actuellement définie de manière itérative, et il serait intéressant de proposer un modèle déclaratif de RCA. Un tel modèle ouvrirait des pistes pour combiner RCA avec d'autres extensions de l'AFC et ainsi étudier leur interopérabilité et coopération. Concrètement, un modèle déclaratif de RCA permettrait de développer de nouvelles extensions de l'AFC qui combine RCA et d'autres extensions existantes. Par exemple, on pourrait imaginer une extension combinant RCA et les structures de motifs (*pattern structures*) [Ganter et Kuznetsov, 2001], afin de tirer parti des avantages des deux extensions, à l'instar de l'extension Graph-PS [Ferré, 2023], qui combine GCA et les structures de motifs. Une piste pour définir un modèle déclaratif de RCA pourrait passer par la construction d'un graphe de descriptions d'une TBox [Baader, 2003], représentant une abstraction du graphe de dépendances d'une famille de treillis RCA, sans inclure les extensions des concepts ni la relation de subsomption. On pourrait également définir l'intension d'un concept RCA en s'inspirant du PGP simulant un concept RCA et du travail de [Kötters, 2016], qui assimile les intensions de concepts RCA aux motifs d'arbres enracinés (*rooted tree patterns*).

Publications

— Vanessa Fokou, Peggy Cellier, Xavier Dolques, Sébastien Ferré, and Florence Le Ber. *Theoretical Comparison of Relational Concept Analysis (RCA) and Graph-FCA (GCA).* International Journal of Approximate Reasoning, 109496, 2025.

— Vanessa Fokou, Peggy Cellier, Xavier Dolques, Sébastien Ferré, and Florence Le Ber. *Relational Concept Analysis and Graph-FCA in Practice : Analyzing Multi-relational Data.*
International Joint Conference on Conceptual Knowledge Structures. Springer. 2025, p. 35-50.

— Vanessa Fokou, Peggy Cellier, Xavier Dolques, Sébastien Ferré, and Florence Le Ber. *Comparing Relational Concept Analysis and Graph-FCA on their Common Ground.* International Joint Conference on Conceptual Knowledge Structures. Springer. 2024, p. 63-79.

— Vanessa Fokou, Karim El Haff, Agnès Braud, Xavier Dolques, Florence Le Ber, and Véronique Pitchon.
Exploring Old Arabic Remedies with Formal and Relational Concept Analysis.
International Joint Conference on Conceptual Knowledge Structures. Springer. 2024, p. 302-318.

Bibliographie

Agrawal, R., Imieliński, T., & Swami, A. Mining association rules between sets of items in large databases. In : In *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*. 1993, 207-216 (cf. p. 1).

Alam, M., Chekol, M. W., Coulet, A., Napoli, A., & Smaïl-Tabbone, M. Lattice based data access (lbda): An approach for organizing and accessing linked open data in biology. In : In *DMoLD'13-Data Mining on Linked Data Workshop (ECML/PKDD, 2013)*. Springer. 2013 (cf. p. 26).

Alam, M., Coulet, A., Napoli, A., & Smaïl-Tabbone, M. Formal concept analysis applied to transcriptomic data. In : In *What can FCA do for Artificial Intelligence (FCA4AI)(ECAI 2012)*. 2012 (cf. p. 8).

Azmeh, Z., Huchard, M., Napoli, A., Hacene, M. R., & Valtchev, P. Querying Relational Concept Lattices. In : In *CLA. 11*. Citeseer. 2011, 377 (cf. p. 81-83).

Baader, F. Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In : In *IJCAI. 3*. Citeseer. 2003, 319-324 (cf. p. 170).

Baader, F., Calvanese, D., Mcguinness, D., Nardi, D., & Patel-Schneider, P. (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge university press. (Cf. p. 32, 89).

Barbut, M., & Monjardet, B. (1970). *Ordre et classification: algèbre et combinatoire*. Hachette. (Cf. p. 1, 8, 10).

Bazin, A., Galasso, J., & Kahn, G. (2024). Polyadic relational concept analysis. *International Journal of Approximate Reasoning*, *164*, 109067 (cf. p. 2, 23, 76).

Belohlavek, R., Sigmund, E., & Zacpal, J. (2011). Evaluation of IPAQ questionnaires supported by formal concept analysis. *Information Sciences*, *181*(10), 1774-1786 (cf. p. 8).

Belohlávek, R., & Vychodil, V. What is a fuzzy concept lattice? In : In *International Conference on Concept Lattices and their Applications*. 2005. https://api.semanticscholar.org/CorpusID:12185007 (cf. p. 21, 23).

Bendaoud, R., Hacene, A. M. R., Toussaint, Y., Delecroix, B., & Napoli, A. Text-based ontology construction using relational concept analysis. In : In *International Workshop on Ontology Dynamics-IWOD 2007*. 2007 (cf. p. 26).

Berry, A., Huchard, M., Napoli, A., & Sigayret, A. Hermes: an Efficient Algorithm for Building Galois Sub-hierarchies. In : In *CLA. 2012*. 2012, 21-32 (cf. p. 77, 100).

Birkhoff, G. (1940). *Lattice theory* (T. 25). American Mathematical Soc. (Cf. p. 1, 8, 10).

Boffa, S. (2022). Extracting concepts from fuzzy relational context families. *IEEE Transactions on Fuzzy Systems*, *31*(4), 1202-1213 (cf. p. 23, 76).

Bordat, J.-P. (1986). Calcul pratique du treillis de Galois d'une correspondance. *Mathématiques et Sciences humaines*, *96*, 31-47 (cf. p. 18).

Braud, A., Dolques, X., Huchard, M., & Le Ber, F. (2018). Generalization effect of quantifiers in a classification based on relational concept analysis. *Knowledge-based systems*, *160*, 119-135 (cf. p. 33, 37, 38, 100, 142, 165, 169).

Bruno, M., Canfora, G., Di Penta, M., & Scognamiglio, R. An approach to support web service classification and annotation. In : In *2005 IEEE International Conference on e-Technology, e-Commerce and e-Service*. IEEE. 2005, 138-143 (cf. p. 8).

Carbonnel, J., Huchard, M., & Nebut, C. Exploring the variability of interconnected product families with relational concept analysis. In : In *Proceedings of the 23rd International Systems and Software Product Line Conference-Volume B*. 2019, 199-206 (cf. p. 26).

Casas-Garriga, G. Summarizing Sequential Data with Closed Partial Orders. In : In *SIAM International Conference on Data Mining*. SDM. 2005 (cf. p. 83-85).

Chein, M. (1969). Algorithme de recherche des sous-matrices premières d'une matrice. *Bulletin mathématique de la Société des Sciences Mathématiques de la République Socialiste de Roumanie*, 21-25 (cf. p. 18).

Chein, M., & Mugnier, M.-L. (2008). *Graph-based knowledge representation: computational foundations of conceptual graphs*. Springer Science & Business Media. (Cf. p. 1, 22).

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 (cf. p. 22).

Demko, C., Boukhetta, S. E., Richard, J., Savarit, G., Bertet, K., Faucher, C., & Mondou, D. GALACTIC: towards a generic and scalable platform for complex and heterogeneous data using Formal Concept Analysis. In : In *CLA*. 2022, 191-198 (cf. p. 2, 21).

Dicky, H., Dony, C., Huchard, M., & Libourel, T. ARES, un algorithme d'Ajout avec REStructuration dans les hiérarchies de classes. In : In *LMO. 94*. 1994, 125-136 (cf. p. 17, 100).

Dicky, H., Dony, C., Huchard, M., & Libourel, T. ARES, Adding a class and REStructuring Inheritance Hierarchy. In : In *BDA*. Citeseer. 1995, 25-42 (cf. p. 43).

Dolques, X., Braud, A., Grac, C., & Le Ber, F. Analyzing water monitoring data with RCA-based approaches. In : In *Workshop RealDataFCA'2021*. 2021 (cf. p. 1, 26).

Dolques, X., Braud, A., Huchard, M., & Le Ber, F. Rcaexplore, a FCA based tool to explore relational data. In : In *Workshop" Applications and Tools of Formal Concept Analysis"(ICFCA)*. 2019 (cf. p. 2, 14, 36, 78, 160).

Dolques, X., Le Ber, F., Huchard, M., & Nebut, C. Relational concept analysis for relational data exploration. In : In *Advances in Knowledge Discovery and Management: Volume 5*. Springer, 2015, p. 57-77 (cf. p. 77).

Džeroski, S. (2003). Multi-relational data mining: An introduction. *SIGKDD Explorations*, *5*, 1-16 (cf. p. 89).

Džeroski, S. Relational data mining. In : In *Data mining and knowledge discovery handbook*. Springer, 2010, p. 887-911 (cf. p. 2).

Ebner, M., Mühlburger, H., Schaffert, S., Schiefner, M., Reinhardt, W., & Wheeler, S. Getting granular on twitter: tweets from a conference and their limited usefulness for non-participants. In : In *IFIP International Conference on Key Competencies in the Knowledge Society*. Springer. 2010, 102-113 (cf. p. 8).

Eklund, P., Ducrou, J., & Brawn, P. Concept lattices for information visualization: Can novices read line-diagrams? In : In *International Conference on Formal Concept Analysis*. Springer. 2004, 57-73 (cf. p. 8).

Even, S. (2011). *Graph algorithms*. Cambridge University Press. (Cf. p. 86, 131).

Falk, I., Gardent, C., & Lorenzo, A. Using formal concept analysis to acquire knowledge about verbs. In : In *7th International Conference on Concept Lattices and Their Applications-CLA 2010*. 2010, 12 (cf. p. 8).

Ferré, S. A proposal for extending formal concept analysis to knowledge graphs. In : In *Formal Concept Analysis: 13th International Conference, ICFCA 2015, Nerja, Spain, June 23-26, 2015, Proceedings 13*. Springer. 2015, 271-286 (cf. p. 2, 23, 52, 76).

Ferré, S. (2019). User Manual of the gfca Tool for the Generation and Visualization of Graph-FCA Concepts (cf. p. 2, 66, 100).

Ferré, S. Exploring the Application of Graph-FCA to the Problem of Knowledge Graph Alignment. In : In *Concept Lattices and Applications*. 2022 (cf. p. 52).

Ferré, S. Graph-FCA Meets Pattern Structures. In : In *International Conference on Formal Concept Analysis*. Springer. 2023, 33-48 (cf. p. 2, 23, 76, 170).

Ferré, S., & Cellier, P. Graph-FCA in practice. In : In *Graph-Based Representation and Reasoning: 22nd International Conference on Conceptual Structures, ICCS 2016, Annecy, France, July 5-7, 2016, Proceedings 22*. Springer. 2016, 107-121 (cf. p. 52, 54).

Ferré, S., & Cellier, P. How Hierarchies of Concept Graphs Can Facilitate the Interpretation of RCA Lattices? In : In *CLA 2018. 2123*. CEUR-WS Proc. 2018 (cf. p. 2, 83, 86-88, 96, 100, 130, 131, 141).

Ferré, S., & Cellier, P. (2020). Graph-FCA: An extension of formal concept analysis to knowledge graphs. *Discrete applied mathematics*, *273*, 81-102 (cf. p. 22, 23, 52, 56, 62, 64, 76, 95, 138).

Ferré, S., & Cellier, P. Modeling Complex Structures in Graph-FCA: Illustration on Natural Language Syntax. In : In *Existing Tools and Applications for Formal Concept Analysis*. 2022 (cf. p. 52).

Ferré, S., Huchard, M., Kaytoue, M., Kuznetsov, S. O., & Napoli, A. Formal concept analysis: from knowledge discovery to knowledge processing. In : In *A Guided Tour of Artificial Intelligence Research: Volume II: AI Algorithms*. Springer, 2020, p. 411-445 (cf. p. 2, 21).

Ferré, S., & Ridoux, O. A logical generalization of formal concept analysis. In : In *International Conference on Conceptual Structures*. Springer. 2000, 371-384 (cf. p. 2, 21, 76).

Fokou, V., Cellier, P., Dolques, X., Ferré, S., & Le Ber, F. Comparing Relational Concept Analysis and Graph-FCA on their Common Ground. In : In *International Joint Conference on Conceptual Knowledge Structures*. Springer. 2024, 63-79 (cf. p. 106, 153).

Fokou, V., Cellier, P., Dolques, X., Ferré, S., & Le Ber, F. Relational Concept Analysis and Graph-FCA in Practice: Analyzing Multi-relational Data. In : In *International Joint Conference on Conceptual Knowledge Structures*. Springer. 2025, 35-50 (cf. p. 144).

Fokou, V., Cellier, P., Dolques, X., Ferré, S., & Le Ber, F. (2025b). Theoretical comparison of Relational Concept Analysis (RCA) and Graph-FCA (GCA). *International Journal of Approximate Reasoning*, 109496 (cf. p. 116, 125, 167).

Fokou, V., El Haff, K., Braud, A., Dolques, X., Le Ber, F., & Pitchon, V. Exploring Old Arabic Remedies with Formal and Relational Concept Analysis. In : In *Int. Joint Conf. Concepts 24*. Springer, 2024, 302-318 (cf. p. 158).

Ganter, B., & Kuznetsov, S. O. Pattern structures and their projections. In : In *International conference on conceptual structures*. Springer. 2001, 129-142 (cf. p. 2, 21, 23, 76, 170).

Ganter, B., & Wille, R. (1999). *Formal Concept Analysis: Mathematical Foundations*. Springer. (Cf. p. 1, 8, 21, 39, 76, 95).

Giscard, P.-L., Rochet, P., & Wilson, R. C. (2017). Evaluating balance on social networks from their simple cycles. *Journal of Complex Networks*, *5*(5), 750-775 (cf. p. 153).

Godin, R., & Mili, H. (1993). Building and maintaining analysis-level class hierarchies using galois lattices. *ACM SIGplan Notices*, *28*(10), 394-410 (cf. p. 17, 100, 165).

Godin, R., Missaoui, R., & Alaoui, H. (1995). Incremental concept formation algorithms based on Galois (concept) lattices. *Computational intelligence*, *11*(2), 246-267 (cf. p. 18, 19).

Guesmi, S., Trabelsi, C., & Latiri, C. Community detection in multi-relational bibliographic networks. In : In *International Conference on Database and Expert Systems Applications*. Springer. 2016, 11-18 (cf. p. 26).

Guesmi, S., Trabelsi, C., & Latiri, C. (2016b). CoMRing: a framework for community detection based on multi-relational querying exploration. *Procedia Computer Science*, *96*, 627-636 (cf. p. 26).

Gutierrez, A., Huchard, M., & Martin, P. FCA4J: a Java library for relational concept analysis and formal concept analysis. In : In *ETAFCA 2022-ExistingTools and Applications for Formal Conceptual Analysis Workshop@ CLA2022. 3308*. (Paper 17). CLA Conference Series (cla. inf. upol. cz). 2022, 207-212 (cf. p. 2, 36, 79, 160).

Gutierrez, A., Huchard, M., Martin, P., & Zhang, H. Empowering relational concept analysis using large language model knowledge delivery. In : In *International Joint Conference on Conceptual Knowledge Structures*. Springer. 2025, 124-139 (cf. p. 165).

Hacene, M. R., Napoli, A., Valtchev, P., Toussaint, Y., & Bendaoud, R. Ontology learning from text using relational concept analysis. In : In *2008 International MCETECH Conference on e-Technologies (mcetech 2008)*. IEEE. 2008, 154-163 (cf. p. 26).

Hahn, G., & Tardif, C. Graph homomorphisms: structure and symmetry. In : In *Graph symmetry: algebraic methods and applications*. Springer, 1997, p. 107-166 (cf. p. 56, 61, 137).

Hanika, T., & Hirth, J. (2019). Conexp-Clj-A Research Tool for FCA. *ICFCA (Supplements)*, *2378*(70-75), 346 (cf. p. 2).

Hitzler, P., Krotzsch, M., & Rudolph, S. (2009). *Foundations of semantic web technologies*. Chapman ; Hall/CRC. (Cf. p. 1).

Hlad, N., Lemoine, B., Huchard, M., & Seriai, A.-D. Leveraging relational concept analysis for automated feature location in software product lines. In : In *Proceedings of the 20th ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences*. 2021, 170-183 (cf. p. 26).

Hodo, F., Pranav, S., & Sertkaya, B. Clustering Knowledge Graphs Using Concept Lattices. In : In *Description Logics*. CEUR Workshop Proceedings, 2023 (cf. p. 89).

Huchard, M., Hacene, M. R., Roume, C., & Valtchev, P. (2007). Relational concept discovery in structured datasets. *Annals of Mathematics and Artificial Intelligence*, *49*, 39-76 (cf. p. 26, 76).

Huchard, M., Martin, P., Muller, E., Poncelet, P., Raveneau, V., & Sallaberry, A. (2024). RCAviz: Exploratory search in multi-relational datasets represented using relational concept analysis. *IJAR*, *166*, 109123 (cf. p. 79, 100, 165, 170).

Huchard, M., Roume, C., & Valtchev, P. When concepts point at other concepts: the case of UML diagram reconstruction. In : In *Proceedings of the 2nd Workshop on Advances in Formal Concept Analysis for Knowledge Discovery in Databases (FCAKDD)*. 2002, 32-43 (cf. p. 26).

Jones, T. H., & Song, I.-Y. (2000). Binary equivalents of ternary relationships in entity-relationship modeling: A logical decomposition approach. *Journal of Database Management (JDM)*, *11*(2), 12-19 (cf. p. 89).

Kahl, O. (2009). *Sābūr ibn Sahl's Dispensatory in the Recension of the 'Aḍudī Hospital*. Brill. (Cf. p. 3, 158).

Kaytoue, M., Codocedo, V., Buzmakov, A., Baixeries, J., Kuznetsov, S. O., & Napoli, A. Pattern structures and concept lattices for data mining and knowledge processing. In : In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2015, 227-231 (cf. p. 2).

Kaytoue, M., Duplessis, S., Kuznetsov, S. O., & Napoli, A. Two fca-based methods for mining gene expression data. In : In *International Conference on Formal Concept Analysis*. Springer. 2009, 251-266 (cf. p. 8).

Keip, P., Ferré, S., Gutierrez, A., Huchard, M., Silvie, P., & Martin, P. Practical comparison of FCA extensions to model indeterminate value of ternary data. In : In *CLA 2020. 2668*. CEUR-WS Proc. 2020, 197-208 (cf. p. 2, 89-91, 101, 146, 148).

Keip, P., Gutierrez, A., Huchard, M., Le Ber, F., Sarter, S., Silvie, P., & Martin, P. Effects of Input Data Formalisation in Relational Concept Analysis for a Data Model with a Ternary Relation. In : In *ICFCA 2019*. 2019, 191-207 (cf. p. 89, 90, 101, 105, 144, 146).

Keller, B. J., Eichinger, F., & Kretzler, M. (2012). Formal concept analysis of disease similarity. *AMIA Summits on Translational Science Proceedings*, *2012*, 42 (cf. p. 8).

Kis, L. L., Sacarea, C., & Troanca, D. (2016). FCA Tools Bundle-A Tool that Enables Dyadic and Triadic Conceptual Navigation. *FCA4AI@ ECAI*, *1703*, 42-50 (cf. p. 91).

Klamt, S., & von Kamp, A. (2009). Computing paths and cycles in biological interaction graphs. *BMC bioinformatics*, *10*(1), 181 (cf. p. 153).

Kötters, J. Concept lattices of a relational structure. In : In *Conceptual Structures for STEM Research and Education: ICCS 2013, Proc.* Springer, 2013, 301-310 (cf. p. 2, 22, 23, 76).

Kötters, J. Intension graphs as patterns over power context families. In : In *Proceedings of CLA. 1624*. 2016, 203-216 (cf. p. 22, 170).

Kuznetsov, S. O. (2007). On stability of a formal concept. *Annals of Mathematics and Artificial Intelligence*, *49*(1), 101-115 (cf. p. 17).

Kuznetsov, S. O., & Makhalova, T. (2018). On interestingness measures of formal concepts. *Information Sciences*, *442*, 202-219 (cf. p. 17).

Lehmann, F., & Wille, R. A triadic approach to formal concept analysis. In : In *Conceptual Structures: Applications, Implementation and Theory: ICCS'95, Proc.* Springer, 1995, 32-43 (cf. p. 2, 21, 89).

Leutwyler, N., Lezoche, M., Panetto, H., & Torres, D. Extending RCA algorithm to consider ternary relations. In : In *12th International Conference on Information Society and Technology*. Eventiotic. 2022, 203-209 (cf. p. 91).

Lounkine, E., Auer, J., & Bajorath, J. (2008). Formal concept analysis for the identification of molecular fragment combinations specific for active and highly potent compounds. *Journal of medicinal chemistry*, *51*(17), 5342-5348 (cf. p. 8).

Messai, N., Devignes, M.-D., Napoli, A., & Smail-Tabbone, M. Querying a bioinformatic data sources registry with concept lattices. In : In *International Conference on Conceptual Structures*. Springer. 2005, 323-336 (cf. p. 81).

Muller, E., Huchard, M., Martin, P., Poncelet, P., & Sallaberry, A. RCAviz: Visualizing and exploring relational conceptual structures. In : CEUR-WS. 2022 (cf. p. 79).

Munzner, T. (2009). A nested model for visualization design and validation. *IEEE transactions on visualization and computer graphics*, *15*(6), 921-928 (cf. p. 79).

Nica, C., Braud, A., Dolques, X., Huchard, M., & Le Ber, F. Extracting Hierarchies of Closed Partially-Ordered Patterns using Relational Concept Analysis. In : In *ICCS 2016*. LNCS vol 9717. Springer, 2016, 17-30 (cf. p. 2, 83-85, 87, 96, 100, 133).

Nica, C., Braud, A., Dolques, X., Huchard, M., & Le Ber, F. Exploring temporal data using relational concept analysis: An application to hydroecology. In : In *CLA: Concept Lattices and their Applications. 1624*. 2016, 299-311 (cf. p. 1, 26).

Nica, C., Braud, A., Dolques, X., Huchard, M., & Le Ber, F. L'analyse relationnelle de concepts pour la fouille de données temporelles–Application à l'étude de données hydroécologiques. In : In *EGC: Extraction et Gestion des Connaissances*. Hermann-Éditions. 2016, 267-278 (cf. p. 26, 133).

Nica, C., Braud, A., & Le Ber, F. Exploring heterogeneous sequential data on river networks with relational concept analysis. In : In *International Conference on Conceptual Structures*. Springer. 2018, 152-166 (cf. p. 85).

Nica, C., Braud, A., & Le Ber, F. (2020). RCA-SEQ: an original approach for enhancing the analysis of sequential data based on hierarchies of multilevel closed partially-ordered patterns. *DAM*, *273*, 232-251 (cf. p. 2, 83, 84, 87).

Nourine, L., & Raynaud, O. (1999). A fast algorithm for building lattices. *Information processing letters*, *71*(5-6), 199-204 (cf. p. 18).

Osswald, R., & Petersen, W. A logical approach to data-driven classification. In : In *Annual Conference on Artificial Intelligence*. Springer. 2003, 267-281 (cf. p. 17).

Ouzerdine, A., Braud, A., Dolques, X., Huchard, M., & Le Ber, F. Adjusting the Exploration Flow in Relational Concept Analysis: An Experience on a Watercourse Quality Dataset. In : In *AKDM*. T. 9. Springer, 2022, p. 175-198 (cf. p. 78).

Petersen, W. (2001). A set-theoretical approach for the induction of inheritance hierarchies. *Electronic Notes in Theoretical Computer Science*, *53*, 296-308 (cf. p. 17).

Poelmans, J., Ignatov, D. I., Kuznetsov, S. O., & Dedene, G. (2013). Formal concept analysis in knowledge processing: A survey on applications. *Expert systems with applications*, *40*(16), 6538-6560 (cf. p. 2, 8, 21).

Poelmans, J., Ignatov, D. I., Kuznetsov, S. O., & Dedene, G. (2014). Fuzzy and rough formal concept analysis: a survey. *International Journal of General Systems*, *43*(2), 105-134 (cf. p. 21).

Prediger, S., & Wille, R. The lattice of concept graphs of a relationally scaled context. In : In *International Conference on Conceptual Structures*. Springer. 1999, 401-414 (cf. p. 22).

Priss, U., & Old, L. J. (2004). Modelling lexical databases with formal concept analysis. (cf. p. 8).

Rouane-Hacene, M., Huchard, M., Napoli, A., & Valtchev, P. (2013). Relational concept analysis: mining concept lattices from multi-relational data. *Annals of Mathematics and Artificial Intelligence*, *67*, 81-108 (cf. p. 2, 22, 26, 27, 29, 33, 43, 48, 76, 87, 91, 95, 165).

Rouane-Hacene, M., Toussaint, Y., & Valtchev, P. Mining safety signals in spontaneous reports database using concept analysis. In : In *Conference on Artificial Intelligence in Medicine in Europe*. Springer. 2009, 285-294 (cf. p. 1, 8).

Saquer, J., & Deogun, J. S. (2001). Concept approximations based on rough sets and similarity measures. *International Journal of Applied Mathematics and Computer Science*, *11*(3), 655-674 (cf. p. 17).

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2008). The graph neural network model. *IEEE transactions on neural networks*, *20*(1), 61-80 (cf. p. 1).

Sedlmair, M., Meyer, M., & Munzner, T. (2012). Design study methodology: Reflections from the trenches and the stacks. *IEEE transactions on visualization and computer graphics*, *18*(12), 2431-2440 (cf. p. 79).

Sklenar, V., Zacpal, J., & Sigmund, E. Evaluation of IPAQ questionnaire by FCA. In : In *CLA*. Citeseer. 2005, 60-69 (cf. p. 8).

Song, I.-Y., & Jones, T. H. Ternary relationship decomposition strategies based on binary imposition rules. In : In *11th ICDE*. IEEE. 1995, 485-492 (cf. p. 89).

Sowa, J. F. (1984). *Conceptual structures: information processing in mind and machine*. Addison-Wesley Longman Publishing Co., Inc. (Cf. p. 1, 22).

Sridharan, G. V., Ullah, E., Hassoun, S., & Lee, K. (2015). Discovery of substrate cycles in large scale metabolic networks using hierarchical modularity. *BMC Systems Biology*, *9*(1), 5 (cf. p. 153).

Stumme, G., Taouil, R., Bastide, Y., & Lakhal, L. (2001). Conceptual clustering with iceberg concept lattices. *Proc. of GI-Fachgruppentreffen Maschinelles Lernen*, *1* (cf. p. 17).

Stumme, G., Taouil, R., Bastide, Y., Pasquier, N., & Lakhal, L. (2002). Computing iceberg concept lattices with Titanic. *Data & Knowl. Eng.*, *42*(2), 189-222 (cf. p. 17, 43, 77, 100, 160, 165).

Stumpfe, D., Lounkine, E., & Bajorath, J. Molecular test systems for computational selectivity studies and systematic analysis of compound selectivity profiles. In : In *Chemoinformatics and computational chemical biology*. Springer, 2010, p. 503-515 (cf. p. 8).

Valtchev, P., & Missaoui, R. Building concept (Galois) lattices from parts: generalizing the incremental methods. In : In *Conceptual Structures: Broadening the Base: 9th International Conference on Conceptual Structures, ICCS 2001 Stanford, CA, USA, July 30–August 3, 2001 Proceedings 9*. Springer, 2001, 290-303 (cf. p. 18, 19).

Van Der Merwe, D., Obiedkov, S., & Kourie, D. Addintent: A new incremental algorithm for constructing concept lattices. In : In *Concept Lattices: Second International Conference on Formal Concept Analysis, ICFCA 2004, Sydney, Australia, February 23-26, 2004. Proceedings 2*. Springer. 2004, 372-385 (cf. p. 18, 19).

Villerd, J., Toussaint, Y., & Louët, A. L.-L. Adverse drug reaction mining in pharmacovigilance data using formal concept analysis. In : In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2010, 386-401 (cf. p. 8).

Voutsadakis, G. (2002). Polyadic concept analysis. *Order*, *19*, 295-304 (cf. p. 2, 22, 23).

Wajnberg, M., Valtchev, P., Lezoche, M., Masse, A. B., & Panetto, H. Concept analysis-based association mining from linked data: a case in industrial decision making. In : In *2nd International Workshop on Data meets Applied Ontologies in Open Science and Innovation, DAO-SI 2019. 2518*. 2019 (cf. p. 26).

Wajnberg, M., Valtchev, P., Lezoche, M., Panetto, H., & Blondin Massé, A. Mining process factor causality links with multi-relational associations. In : In *Proceedings of the 10th International Conference on Knowledge Capture*. 2019, 263-266 (cf. p. 26).

Wille, R. (1982). Restructuring lattice theory; an approach based on hierarchies of concept. *Ordered Sets/Reidel* (cf. p. 8).

Wille, R. Conceptual graphs and formal concept analysis. In : In *Conceptual Structures: Fulfilling Peirce's Dream: Fifth International Conference on Conceptual Structures, ICCS'97 Seattle, Washington, USA, August 3–8, 1997 Proceedings 5*. Springer. 1997, 290-303 (cf. p. 2, 22, 102).

Wille, R. Existential concept graphs of power context families. In : In *International Conference on Conceptual Structures*. Springer. 2002, 382-395 (cf. p. 22).

Vanessa Laure FOKOU

Comparaison empirique et théorique d'approches en analyse de concepts formels pour les données multi-relationnelles

Résumé : L'Analyse Relationnelle de Concepts (RCA) et Graph-FCA (GCA) sont deux extensions majeures de l'Analyse Formelle de Concepts développées pour le traitement des données multi-relationnelles. Ce travail propose une étude comparative à la fois théorique et empirique, des deux approches RCA et GCA dans le but d'établir leurs similitudes et différences sur une base solide, et de fournir à l'analyste des repères pour choisir l'approche la plus adaptée selon la nature des données et des objectifs d'analyse. Nous démontrons que GCA est plus expressif que RCA en montrant que l'ensemble de concepts produits par RCA est inclus dans l'ensemble de concepts produits par GCA. Nous proposons également une transformation de la famille de treillis RCA en un ensemble de graphes, permettant ainsi d'améliorer l'exploration et l'interprétation des résultats. Une mise en œuvre sur un jeu de données issu d'une ancienne pharmacopée arabe a permis de mettre en avant la complémentarité de ces deux approches.

Mots clés : Analyse Formelle de Concepts, Analyse Relationnelle de Concepts, Graph-FCA, Treillis, Fouille de données multi-relationnelles, Graphe de connaissances, Motif de graphe.

Abstract : Relational Concept Analysis (RCA) and Graph-FCA (GCA) are two major extensions of Formal Concept Analysis developed for the processing of multi-relational data. This work proposes a comparative study, both theoretical and empirical, of the two approache, RCA and GCA, with the aim of establishing their similarities and differences on a solid basis, and providing analysts with guidelines for choosing the most appropriate approach according to the nature of the data and the objectives of the analysis. We demonstrate that GCA is more expressive than RCA by showing that the set of concepts produced by RCA is included in the set of concepts produced by GCA. We also propose a transformation of the RCA lattice family into a set of graphs, thus improving the exploration and interpretation of results. An experiment on a dataset from an ancient Arabic pharmacopoeia highlighted the complementarity of these two approaches.

Keywords : Formal Concept Analysis, Relational Concept Analysis, Graph-FCA, Lattice, Multi-relational data mining, Knowledge graph, Graph pattern.