THÈSE POUR OBTENIR LE GRADE DE DOCTEUR DE L'UNIVERSITE DE MONTPELLIER

En Informatique

École doctorale : Information, Structures, Systèmes

Unité de recherche AIDA, CIRAD

Conversion automatique de modèles et de jeux de données pour l'exploration conceptuelle : Application à une base de connaissances du vivant

Présentée par Priscilla Keip

Le 16 Décembre 2021

Sous la direction de Marianne Huchard

Devant le jury composé de

Mario Lezoche, Maître de Conférences, CRAN, Université de Lorraine, Nancy	Rapporteur
Isabelle Wattiau, Professeur, Dpt SI/Décision/Stat., ESSEC Business School, Paris	Rapportrice
Thierry Bourgoin, Professeur, ISYEB, Muséum National d'Histoire Naturelle, Paris	Examinateur
Isabelle Mougenot, Maître de Conférences, Espace-Dev, Univ. Montpellier, Montpellier	Examinatrice
Farouk Toumani, Professeur, LIMOS, Univ. Blaise-Pascal, Clermont-Ferrand	Président du jury
Marianne Huchard, Professeur, LIRMM, Univ. Montpellier, Montpellier	Directrice
Pierre Martin, Chercheur, AÏDA, CIRAD, Montpellier	Co-encadrant
Pierre Silvie, Chercheur, AÏDA et PHIM, IRD, Montpellier	Co-encadrant

UNIVERSITÉ DE MONTPELLIER

Résumé

Les Sciences du vivant et de l'environnement génèrent de nombreuses bases de données et de connaissances. L'exploration conceptuelle est une approche de fouille de données qui permet d'en extraire de nouvelles connaissances. Les méthodes d'exploration conceptuelle considérées dans ce travail relèvent de l'Analyse de Concepts Formels (FCA). Toute méthode issue de FCA imposant en entrée une structuration prédéfinie des données, la question de recherche traitée concerne la conversion automatique d'une base de données ou de connaissances en vue de son exploration avec FCA ou avec l'Analyse de Concepts Relationnels (RCA), une extension aux données multi-relationnelles.

Pour asseoir ce travail, nous avons utilisé la base de connaissances Knomana, qui rassemble des descriptions d'usage de plantes. Selon les connaissances considérées, diverses problématiques de conversion doivent être résolues pour permettre une fouille de données pertinente par les utilisateurs finaux. Ces problématiques peuvent se situer au niveau du schéma (modèle de classes) ou au niveau des instances, principalement (1) la conversion des relations de spécialisation en aplatissant la hiérarchie d'héritage; (2) la conversion d'une relation ternaire via la matérialisation ou des relations binaires, et (3) la levée de l'indétermination de la désignation d'organismes vivants par l'usage de l'abréviation spp. dans sa dénomination binominale (linnéenne).

La méthodologie adoptée relève de l'ingénierie dirigée par les modèles, de la transformation des bases de données, du refactoring en ingénierie logicielle et de la conversion d'ontologies.

Cette thèse présente un algorithme général de conversion d'un jeu de connaissances pour RCA, dont la structure est représentée sous la forme d'un diagramme de classes au format UML. L'algorithme est appliqué à Knomana et diverses modélisations de la relation ternaire en relations binaires et une modélisation de la levée de l'indétermination d'un nom d'espèce sont expérimentées. L'évaluation de cet algorithme est effectuée en regard de son applicabilité, de son aptitude à être utilisé pour des volumes de données importants (i.e. passage à l'échelle), et de sa pertinence. La perspective offerte par ce travail est de permettre l'analyse de tout jeu de données et de connaissances avec FCA et RCA.

Mots clefs : Base de connaissances, Base de données, Exploration conceptuelle, Analyse de Concepts Formels, Conversion, Knomana, Relation n-aire, Indétermination, Ingénierie dirigée par les modèles

Remerciements

Tout d'abord, je souhaite remercier le CIRAD et l'Institut Convergences Agriculture Numérique #DigitAg car sans leur financement, cette thèse n'aurait jamais pu voir le jour. Je remercie également les membres de l'unité AIDA qui m'ont accueillie parmi eux et m'ont soutenue durant cette période.

Je souhaite remercier également les différentes personnes ayant accepté d'évaluer mon travail en tant que membres des comités de suivi individuels ou en tant que membres du jury lors de la soutenance. Je remercie particulièrement Monsieur Farouk Toumani d'avoir présidé le jury. Je remercie également Monsieur Mario Lezoche et Madame Isabelle Wattiau d'avoir accepté de rapporter ma thèse. De même, je remercie Monsieur Thierry Bourgoin et Madame Isabelle Mougenot d'avoir accepté d'examiner mon travail de thèse.

Ce travail n'aurait pas été le même sans la collaboration de nombreuses personnes dont notamment Florence Le Ber et Sébastien Ferré. De même, je remercie Alain Gutierrez pour tout le travail technique effectué, sa réactivité et sa patience pendant qu'il faisait tourner toutes les expérimentations sur son ordi et dont les photos des petites bêtes qui vagabondent dans son jardin seront toujours les bienvenues.

Moins techniques certes, mais tout aussi importantes, je remercie également les secrétaires du CIRAD et du LIRMM, Christine, Jocelyne, Sabrina et Gladys qui répondaient toujours présentes dès que j'avais besoin de quelque chose.

Et enfin, les meilleurs pour la fin. Je remercie profondément ma directrice, Marianne Huchard pour son implication non seulement professionnelle mais aussi et surtout personnelle. Son soutien dans tous les domaines aura été la clef de voûte de ces quatre années de travail commun. Au fil du temps, elle est devenue un peu comme une deuxième maman à qui je pouvais confier tous mes doutes et toutes mes angoisses ce qui a souvent permis de débloquer certaines situations, surtout en cette période difficile où tout se faisait à distance. Merci également à mes deux co-encadrants, Pierre Martin et Pierre Silvie pour leur nombreux conseils m'ayant permis de m'améliorer et dont le sens de l'humour était malheureusement souvent incompris.

Et évidement, même si je ne lui dis pas assez, merci Anthony qui me supporte tous les jours depuis mon arrivée à Montpellier il y a 8 ans ♡

Sommaire

Introduction **1**

1 Construction et formulation de la problématique **11**

1.1 La base de connaissances Knomana 11

1.1.1 Exemple de jeu de connaissances caractéristique - PPAf 13

1.1.2 Les dictionnaires 16

1.2 Motivations à utiliser l'Analyse de Concepts Formels pour explorer Knomana 18

1.3 Proposition d'architecture logicielle pour Knomana 21

1.3.1 Exemples d'utilisation de Knomana 21

1.3.2 Architecture logicielle envisagée 22

1.3.3 Langage de modélisation interne des données 24

1.4 Problématique de la thèse 25

2 Analyse de Concepts Formels et Relationnels **27**

2.1 Analyse de Concepts Formels (FCA) 27

2.2 Les structures conceptuelles de FCA 29

2.3 Analyse de Concepts Relationnels (RCA) 32

2.4 Triadic concept analysis (TCA) 35

2.5 Analyse conceptuelle de graphes de connaissance (Graph-FCA ou GFCA) . 36

2.6 Scaling 37

2.7 Conclusion 39

3 État de l'art sur la conversion des connaissances 41

3.1 Conversion dans le domaine des bases de données 41

3.2 Le refactoring en Génie Logiciel 43

3.3 La construction de jeux de données en science des données 44

3.4 La conversion appliquée au domaine de la représentation des connaissances 45

3.4.1 Conversion et extraction d'une ontologie 45

3.4.2 Préparation des données pour l'Analyse de Concepts Formels 47

3.5 Synthèse 48

3.6 Conclusion 51

4 Méthodes de conversion des connaissances 53

4.1 Conversion depuis un diagramme de classes UML vers une famille de contextes relationnels (RCF) 53

4.1.1 Étape 1 : Transformation des éléments du diagramme non compatibles 57

4.1.2 Étape 2 : Génération des attributs booléens (via *scaling* des attributs multi-valués) 65

4.1.3 Étape 3 : Conversion du diagramme de classes UML modifié vers une RCF 66

4.2 Diverses modélisations de la relation ternaire 68

4.3 Modélisation des données indéterminées 69

4.4 Exemples caractéristiques des transformations de modèles UML en RCF . 70

4.4.1 Exemple de conversion d'une association n-aire 72

4.4.2 Exemple de conversion de l'héritage 74

4.4.3 Exemple de conversion pour des données indéterminées 74

4.5 Conclusion 76

5 Évaluation des conversions 77

5.1 Définition des critères d'évaluation 78

5.2 Évaluation des conversions 79

5.2.1 Évaluation quantitative pour l'héritage 80
5.2.2 Évaluation quantitative de la conversion d'une association n-aire . . 83
5.2.3 Évaluation quantitative pour une classe d'association 88
5.3 Évaluation des structures conceptuelles 90
5.3.1 Évaluation quantitative 90
5.3.2 Évaluation qualitative 95
5.3.3 Modélisation des données indéterminées par FCA et ses extensions . 99
5.4 Évaluation de la validité de l'approche expérimentale 103
5.5 Conclusion 105

6 Conclusion et perspectives **107**
6.1 Résumé des contributions 107
6.2 Perspectives 111

Bibliographie **115**

Introduction

Le contexte de l'agro-écologie L'évolution des préoccupations en agriculture, notamment en lien avec l'impact du changement climatique et la perte de la biodiversité, a conduit les acteurs du monde agricole, du chercheur au technicien, à rechercher des systèmes de production alimentaire durable. Pour répondre à cette ambition, les Nations Unies proposent d'adopter l'agroécologie comme modèle de production agricole. Décrit de façon synthétique, ce modèle intègre les principes de l'écologie dans l'agriculture en conciliant, entre autres, la régénération des ressources naturelles et les souhaits alimentaires des consommateurs. L'agroécologie s'adresse à tous les types de production agricole, de la petite parcelle mélangeant diverses cultures agricoles à la très grande surface n'en comportant qu'une seule, avec le devoir d'adaptation des techniques agricoles pour considérer, par exemple, la diversité de la flore locale et de la faune locale. D'après le Groupe d'experts de haut niveau sur la sécurité alimentaire et la nutrition (HLPE), interface science-politique du Comité des Nations Unies sur la sécurité alimentaire mondiale, l'identification de systèmes de production alimentaires adaptés aux contextes socio-agricoles locaux requiert la coproduction et le partage des connaissances entre scientifiques et producteurs agricoles [HLPE, 2019].

D'autres ambitions sociales se surajoutent à cette préoccupation de production alimentaire durable, à l'exemple de l'approche One Health (`https://onehealthinitiative.com`). Cette approche prône la considération conjointe des santés humaine, animale et écologique, qui sont interdépendantes [Frank, 2008]. Un exemple caractéristique est celui de l'usage excessif des antibiotiques en aquaculture [O'Neill, 2016]. La portée de la réduction de l'emploi des antibiotiques en aquaculture dépasse la simple gestion sanitaire des poissons. En effet, l'administration des antibiotiques aux poissons est effectuée via leur alimentation (e.g. granulés) qui est jetée dans l'eau. Une partie de cette alimentation étant entraînée par le cours d'eau, l'antibiotique se trouve dilué dans l'eau, puis, par la suite, est absorbé par l'être humain et les animaux lorsque l'eau n'est pas préalablement traitée (e.g. bouillie). Le problème est que leur utilisation excessive en aquaculture peut entraîner le développement d'une antibiorésistance par l'être humain [Carvalho and Santos, 2016]. Il s'ensuit que l'antibiotique, pour lequel la bactérie est devenue résistante, ne peut plus être utilisé pour soigner la maladie pour laquelle il a été conçu.

Des solutions alternatives consistent par exemple à utiliser des plantes (locales) pour leur propriétés antibactériennes (e.g. [Reverter et al., 2020]). Les propriétés de ces plantes doivent cependant être connues et maîtrisées, du fait de la toxicité de certaines, de façon à éviter d'engendrer de nouveaux problèmes sanitaires. Tout comme pour l'agroécologie, respecter l'approche One Health requiert le partage de connaissances entre praticiens et

chercheurs de disciplines différentes (médecin, vétérinaire, botaniste, phytopathologiste, etc.).

Le recueil, le partage et l'exploitation de la connaissance Que ce soit pour la mise en œuvre de l'agroécologie ou l'approche One Health, traiter la question du partage de la connaissance demande au préalable de définir ce terme. De nombreux auteurs définissent la notion de connaissance en regard de la pyramide DIKW, l'acronyme de « Data, Information, Knowledge, Wisdom, » proposée par [Ackoff, 1989]. Diverses définitions de cette notion sont données dans la littérature, sans qu'un consensus n'ait été trouvé [Rowley, 2007]. En outre, certains auteurs suggèrent d'ajouter un ou plusieurs niveaux intermédiaires, à l'exemple de [Dammann, 2019] qui propose d'insérer la notion de preuve (« evidence » en anglais) entre Information et Connaissance.

Pour ce mémoire, nous conservons la structure DIKW et proposons la définition synthétique suivante de ces notions : une donnée est une valeur brute, qui ne porte pas de signification intrinsèque ; une information est une donnée à laquelle une signification est donnée ; une connaissance correspond à l'appropriation par un individu ou un groupe d'individus d'une ou de plusieurs informations combinées dans un objectif opérationnel.

Pour partager les connaissances, une solution courante consiste à mettre les connaissances à disposition sur un site web, à l'exemple de la plateforme des connaissances sur l'agroécologie de la FAO (`http://www.fao.org/agroecology/home/fr/`). L'autre solution consiste à capitaliser ces connaissances dans un système à base de connaissances (SBC). Le SBC vise à fournir des méthodes de découverte de connaissances et d'exploration des données afin d'aider à la prise de décision [Fayyad et al., 1996]. Le SBC comporte une base de connaissances, qui de même que pour la base de données, est un recueil de connaissances, pouvant également comporter des informations et des données.

De nombreux SBC en lien avec l'agriculture et l'environnement sont présentés dans la littérature et peuvent être distingués selon deux types, selon qu'ils répondent à une question précise (1er type) ou d'un ordre plus général ou à plusieurs questions (2ème type).

Parmi les SBC du premier type, celui de [Wako, 2017] s'intéresse au diagnostic des dégâts d'une culture de blé provoqués par des bioagresseurs. Pour ce faire, les connaissances, initialement représentées sous forme de réseau sémantique (appelé arbre de décision par les auteurs mais correspondant *de facto* en la représentation graphique des connaissances), sont transformées sous forme de règles logiques (If ... Then ...) puis combinées en utilisant le langage de programmation Prolog. Le SBC de [Yelapure, 2017] permet d'identifier le type de fertilisation à apporter à une culture de soja d'après la composition des feuilles de plantes (teneur en phosphore, etc.) à différents stades de développement de la culture. Pour ce faire, les connaissances sont exprimées sous forme de règles logiques floues, puis combinées par un mécanisme d'inférence. Le SBC de [De Silva and Wickramarachchi, 1998] supervise la découpe de poisson dans un laboratoire. Les connaissances sont exprimées sous forme de règles logiques, et une matrice floue en est dérivée. La supervision consiste à appliquer l'inférence sur la base de règles floues d'après les données d'entrée fournies par les capteurs de mesure pour agir sur les actionneurs.

D'autres SBC du premier type, tels que ceux développées par [Elqassas and Abu-Naser, 2018], [Birhanie and Tegegne, 2019] et [Kumar et al., 2018] cherchent à identifier les maladies sur des arbres fruitiers à partir de photos des symptômes, les deux premiers portant sur le manguier et le dernier sur le bananier. Pour ces trois SBC, les connaissances correspondent à des règles logiques établissant la correspondance entre les symptômes et une probabilité de maladie, la maladie effective étant identifiée par un mécanisme d'inférence. Pour concevoir son SBC, [Elqassas and Abu-Naser, 2018] a utilisé CLIPS, un générateur de système expert [Abu-Naser and Hilles, 2016].

La juxtaposition de toutes ces descriptions montre que les SBC de ce premier type sont structurés de la même façon, i.e. des connaissances exprimées sous forme de règles logiques (If ... Then ...) qui sont combinées par un mécanisme d'inférence. Cette structure correspond à celle adoptée par les systèmes experts classiques, constituée d'une base de faits, d'une base de règle et d'un moteur d'inférence.

Le second type de SBC permet de répondre à une question d'un ordre plus général ou à plusieurs questions. Le SBC de [Rodríguez-García et al., 2021] a pour objet de reconnaître les insectes et maladies sur des cultures agricoles diverses, i.e. l'amandier, l'olivier et la vigne dans sa première version logicielle. La base de connaissances repose sur CropPestO [Rodríguez-García and García-Sánchez, 2020], une ontologie de domaine qui associe des symptômes et des modes de gestion en agriculture conventionnelle et biologique à des insectes et des maladies. Une ontologie est une spécification formelle d'une conceptualisation partagée d'un domaine [Knublauch and Kontokostas, 2017]. Pour le SBC, CropPestO a été automatiquement peuplée via des techniques de NLP (Natural Language Processing) appliquées à des documents non structurés (guides, etc.). Pour procéder à l'identification du ou des insectes et maladies, une analyse par NLP extrait le nom de la culture et les symptômes décrits en langage naturel par l'utilisateur, puis un score d'identité, exprimé en termes de sensibilité et de spécificité, des symptômes observés *versus* ceux décrits dans la base de connaissances, est calculé. En sortie, le SBC indique à l'utilisateur la liste des insectes et maladies, classés d'après leur valeur de score, susceptibles de causer les symptômes observés, ainsi que les modes de gestion associés.
Le SBC de [Coetzer et al., 2014] est un système de médiation visant à améliorer l'interopérabilité sémantique entre les jeux de données issus de trois musées, portant sur les interactions écologiques de spécimens d'insectes visitant des fleurs. La base de connaissances comporte cinq ontologies formalisant les connaissances de domaine : écologie comportementale des visiteurs de fleurs, les visiteurs de fleurs, les visiteurs de fleurs présentant un comportement d'insectes du genre *Rediviva* pour différencier les spécimens qui visitent la fleur pour recueillir son huile ou non, les groupes de différents rangs taxonomiques de visiteurs de fleur, et les connaissances tacites des chercheurs sur l'écologie comportementale. Dans cet exemple de SBC, deux ontologies sont utilisées pour relier les connaissances tacites des chercheurs aux concepts communs de la biodiversité, i.e. darwin-semantic-web [Baskauf and Webb, 2014] et Population and Community Ontology (`http://www.obofoundry.org/ontology/pco.html`). Deux ontologies d'application permettent d'assurer la médiation des données pour les convertir en informations, i.e. le comportement des visiteurs de fleurs et la date d'observation. L'utilisation de Jena, une interface de programmation d'application (API) du langage de programmation Java, établit la correspondance, de façon textuelle, entre les données et les concepts des ontologies. L'utilisation du raisonneur HermiT [Glimm et al., 2014] permet de classer les informa-

tions (i.e. les instances). Une requête exprimée dans le logiciel Protégé, formulée dans une logique de description, concernant par exemple l'embranchement des Arthropodes, peut permettre à un chercheur de connaître les observations des spécimens de cette famille faites dans les trois musées pour en identifier les similitudes et différences d'après son expertise.
Le SBC de [Skobelev et al., 2019] vise à coordonner les décisions de gestion des exploitations agricoles dans un cadre du « smart farming », i.e. l'agriculture intelligente [Wolfert et al., 2017]. La modélisation adoptée repose sur les systèmes multi-agents, où chaque production agricole est représentée par un agent autonome. La base de connaissances liée à ce SBC, implémentée dans chaque agent, décrit une production agricole par plusieurs ontologies : une ontologie de planification, des ontologies descriptives de différents secteurs de la production agricole (i.e. croissance d'une culture, maladie du végétal, type de sol, intervention culturale, machinerie, gestion de l'entreprise, etc.) et un modèle du domaine pour les inter-relier. Ce SBC permet de résoudre les conflits décisionnels émergents dans le pilotage de productions agricoles concurrentes.

Pour les trois SBC de ce second type, les bases de connaissances reposent sur l'utilisation d'ontologies, thème sur lequel de nombreuses recherches sont actuellement conduites et discutées dans le cadre de conférences dédiées, telles que The Conferences on Formal Ontology in Information Systems (FOIS, `https://iaoa.org/fois/`) et Terminology & Ontology: Theories and applications (TOTh, `http://toth.condillac.org`). En revanche, les méthodes de découverte de connaissances et d'exploration des données diffèrent, i.e. un calcul de score d'identité par [Rodríguez-García et al., 2021] ou de proximité lexicale par [Coetzer et al., 2014], ou la modélisation en système multi-agents par [Skobelev et al., 2019].

D'autres technologies d'exploration peuvent être mobilisées à l'exemple de [Nolack Fote et al., 2020] qui, dans sa proposition d'architecture logicielle de SBC pour la prise de décision, dans un contexte de données massives (i.e. basé sur des données longitudinales), indique la possibilité d'utiliser divers types de méthodes de fouille de données telles que la classification, le clustering, ou l'approche statistique. Ainsi, dans le domaine de la santé humaine, et en particulier celui de l'assistance à l'automédication, [Sung and Chi, 2020] propose un SBC dans lequel la base de connaissances comporte une ontologie et utilise une méthode de classification appelée l'Analyse de Concepts Formels [Ganter and Wille, 1999].

Cette thèse s'inscrit dans le contexte des SBC et s'intéresse plus particulièrement à un ensemble de connaissances recueillies sur l'usage des plantes en agriculture, pour répondre à des enjeux de l'agroécologie et de l'approche One Health [Silvie et al., 2021]. Ces connaissances sont organisées dans un SBC appelé *Knomana*. Son architecture logicielle est actuellement au stade de prototype et l'ambition des développeurs de ce SBC est de développer une architecture logicielle dédiée à l'exploration des connaissances qualitatives, non restreintes à celles de Knomana [Keip et al., 2019b] pour explorer des bases de connaissances. Pour l'exploration du SBC Knomana, la méthode de fouille de données privilégiée actuellement est l'Analyse de Concepts Formels, tout comme [Sung and Chi, 2020]. Le projet de recherche global explore cette méthode de fouille pour certaines de ses propriétés telles que la prise en compte de données complexes (catégorielles, symboliques, taxinomiques, relationnelles) ou l'explicabilité des résultats grâce à une interprétation

logique ou textuelle rendue possible par le caractère descriptif des objets et par leur classification.

Positionnement de la thèse Quelle que soit la méthode de fouille de données utilisée, la transformation du jeu de données brutes en données d'entrée de la méthode est un aspect très sensible du processus de découverte de connaissances (processus KDD) [Fayyad et al., 1996]. Dans notre cas, il ne s'agit pas d'un simple changement de format. Plusieurs problèmes se présentent :

- Les données sont parfois incomplètes ou volontairement indéterminées. Il faut donc décider de la manière de traduire cette incomplétude ou de lever cette indétermination.
- Les questions posées par les utilisateurs peuvent varier. Aussi il est important d'avoir une méthode d'extraction des données qui soit flexible.
- Les données répondent à des modèles conceptuels contenant des relations, et notamment une relation ternaire centrale, qui décrit les systèmes de protection. Cette relation associe, pour le dire ici brièvement, un bioagresseur, une culture attaquée et une plante qui peut la protéger contre ce bioagresseur. Les modèles conceptuels contiennent également des relations de spécialisation/généralisation. Ces relations doivent être correctement transformées pour ne pas altérer le jeu de données brutes.
- Comme les modèles conceptuels contiennent des relations, nous visons l'utilisation d'une extension de l'Analyse de Concepts Formels appelée Analyse de Concepts Relationnels [Hacene et al., 2013]. Cependant celle-ci met l'accent sur les relations plus que sur d'autres aspects complexes des données et attend en entrée des ensembles de tableaux binaires, contrairement à d'autres extensions de l'Analyse de Concepts Formels, comme les Pattern Structures [Ganter and Kuznetsov, 2001]. La transformation de données qualitatives ou quantitatives en valeurs binaires doit être correctement effectuée pour ne pas perdre d'informations.

La thèse se positionne comme une contribution à plusieurs problèmes rencontrés dans les premières étapes du processus KDD : **Comment intégrer l'indétermination des données dans le processus ? Comment définir, pour un modèle conceptuel quelconque (UML pouvant être pris comme méta-modèle de référence), un catalogue d'opérations de conversion et un algorithme contrôlant leur application, permettant d'obtenir l'entrée des algorithmes de fouille (calculs de classifications conceptuelles ou de règles) basés sur l'Analyse de Concepts Relationnels ?**

Traiter la seconde question pourrait sembler très dépendant du méta-modèle UML d'un côté, et du modèle d'entrée de l'Analyse de Concepts Relationnels de l'autre. En l'étudiant, nous avons cependant pu observer son caractère assez général. Par exemple, d'autres méthodes de fouille ou de représentation de connaissances utilisant les langages communs du web sémantique (comme OWL ou RDF) se posent en partie les mêmes problèmes, notamment comment transformer une relation ternaire en relations binaires.

Comme autre exemple, passer d'une conception UML à une base de données relationnelles demande de réfléchir à la représentation des relations de spécialisation/généralisation. Enfin, ce travail de conversion demande de convertir des modèles peuplés, c'est-à-dire opérer à la fois au niveau des modèles (ou schémas) et des objets (ou instances).

Organisation du mémoire La suite de ce mémoire est organisée en 6 chapitres.

Le chapitre 1 présente plus en détail le SBC Knomana. Il motive l'utilisation de l'Analyse de Concepts Formels, décrit une architecture pour le SBC et formule en détail la problématique traitée dans la thèse.

Le chapitre 2 décrit l'Analyse de Concepts Formels et l'Analyse de Concepts Relationnels afin de mieux expliquer le type de données attendues en entrée par ces deux méthodes.

Le chapitre 3 présente un ensemble de travaux cherchant à illustrer la manière dont la conversion de modèles ou de schémas, qu'ils soient ou non peuplés, a été traitée dans quatre domaines principaux : en bases de données, en génie logiciel, en science des données et en représentation des connaissances. Cette partie ne se veut pas exhaustive. Son objectif est de montrer à quel point ce problème est général et de recueillir les principales idées développées pour la conversion.

Le chapitre 4 présente le catalogue d'opérations de conversion ainsi qu'un algorithme pour les mettre en œuvre de manière globale. Des exemples de mise en œuvre sont aussi donnés.

Le chapitre 5 évalue ces transformations. Il présente les différents critères d'évaluation, les cas d'étude utilisés et analyse les résultats obtenus. Les analyses sont quantitatives et qualitatives. L'analyse quantitative consiste à examiner les résultats pour les cas d'étude grâce à des métriques. L'analyse qualitative consiste à examiner la pertinence des structures conceptuelles produites à partir du résultat des conversions. Pour ce faire, un des procédés d'analyse consiste à appliquer l'Analyse de Concepts Formels et l'Analyse de Concepts Relationnels pour les cas d'étude.

Enfin, le chapitre 6 conclut le mémoire en rappelant les principales contributions et en développant des travaux à venir.

Collaborations ayant mené à des publications Ce travail a été réalisé au cours de plusieurs collaborations directes qui ont mené à des publications. Elles ont été effectuées avec les personnes suivantes :

- Pierre Silvie, Chercheur à l'IRD, entomologiste, co-porteur du projet Knomana
- Pierre Martin, Chercheur modélisateur au CIRAD, co-porteur du projet Knomana
- Marianne Huchard, Professeur à l'Université de Montpellier
- Sébastien Ferré, Professeur à l'Université de Rennes 1
- Alain Gutierrez, Ingénieur d'études au CNRS
- Florence Le Ber, Directrice de Recherche à l'ENGEES

- Samira Sarter, Directrice de Recherche au Cirad, partenaire du projet Knomana
- Alexandre Bazin, Jessie Carbonnel, Giacomo Kahn, postdoctorants au moment des travaux et Amirouche Ouzerdine, condisciple de Master

Co-encadrements Ce travail a conduit à co-encadrer, en stage ou en TER, plusieurs étudiants sur des travaux connexes. Ces étudiants sont les suivants :

- Cherfaoui Youcef, Martial Jonathan Morel Anthony, Muller Emile, 2020. *Knomana - Extraction d'information dans une base de connaissances sur la santé par les plantes.* TER de Master Informatique, 1ère année. Parcours Données Connaissances et Langage naturel. Cirad, du 09/01 au 20/05/2020. 27p.

- Aougbi Lina, Diboune Saadi Kacioui Arezki, 2020. *Knomana : Extraction d'informations dans une base de connaissances sur la santé par les plantes.* TER de Master Informatique, 1ère année. Parcours Informatique pour les Sciences. Cirad, du 09/01 au 20/05/2020. 32p.

- Mahrach Lina, 2020. Extraction de règles d'une base de connaissances du vivant classée par l'analyse de concepts relationnels. Stage de Master Sciences et Numérique pour la Santé, 2ème année. Parcours Bioinformatique, Connaissances, Données. Cirad, du 02/03 au 31/08/2020. Ce stage a donné lieu à une publication ([Mahrach et al., 2021]).

- Mokhnache Fahima, 2020. Intégration de la relation ternaire dans un logiciel de transformation de modèles UML. Stage de Master Sciences et Numérique pour la Santé, 1ère année. Parcours Bioinformatique, Connaissances, Données. Cirad, du 18/05 au 30/06/2020.

- Mbodj Assane, 2020. Développement d'un requêteur multidimensionnel de connaissances sur la santé animale et végétale. Parcours Bioinformatique, Connaissances, Données. Cirad, du 27/04 au 30/06/2020. 23p.

- Mokhnache Fahima, 2019. Etude de modèles conceptuels pour l'analyse de données sur les plantes à effet pesticide. Stage bibliographique pour le Master Sciences et Numérique pour la Santé, 1ère année. Parcours Bioinformatique, Connaissances, Données. Cirad, du 15/11/2019 au 12/01/2020. 14p.

- Mbodj Assane, 2019. Extraction de règles d'une base de connaissances du vivant classée par l'Analyse de Concepts Formels. Stage bibliographique pour le Master Sciences et Numérique pour la Santé, 1ère année. Parcours Bioinformatique, Connaissances, Données. Cirad, du 15/11/2019 au 12/01/2020. 19p.

- Mahrach Lina, 2019. Contribution à la classification de la base de données Knomana par les méthodes ACF/ACR. Stage de Master Sciences et Numérique pour la Santé, 1ère année. Parcours Bioinformatique, Connaissances, Données. Cirad, du 15/04 au 17/06/2019. 24p.

- Samb Moussa, 2019. Développement d'une interface de Navigation pour la base de connaissances Knomana. Stage de Master Sciences et Numérique pour la Santé,

1ère année. Parcours Bioinformatique, Connaissances, Données. Cirad, du 23/04 au 26/04/2019. 26p.

Ce travail a été financé par le CIRAD et l'Institut Convergences Agriculture Numérique #DigitAg. Il a bénéficié d'une aide de l'État gérée par l'Agence Nationale de la Recherche au titre du programme d'Investissements d'Avenir portant la référence ANR-16-CONV-0004.

Publications Les travaux de thèse ont donné lieu à plusieurs publications dont les références sont rassemblées ci-après :

- Braud A., Dolques X., Gutierrez A., Huchard M., Keip P., Le Ber F., Martin P., Nica C. and Silvie P., 202X. Dealing with Large Volumes of Complex Relational Data using Relational Concept Analysis. Book chapter. To appear in: Editors: Rokia Missaoui, Léonard Kwuida and Talel Abdessalem "Complex Data Analytics with Formal Concept Analysis" (Springer), 28p.
- Silvie P., Martin P., Huchard M., Keip P., Gutierrez A., Sarter S., 2021. Prototyping a knowledge-based system to identify botanical extracts for plant health in Sub-Saharan Africa. Plants, 10 (5), p 896–920
- Mahrach L., Gutierrez A., Huchard M., Keip P., Silvie P., Martin P., 2021. Combining Implication Rules and Conceptual Analysis to Learn from a Pesticidal Plant Knowledge Base. Graph-Based Representation and Reasoning - 26th International Conference on Conceptual Structures, ICCS 2021, September 20-22, 2021. Lecture Notes in Computer Science 12879, Springer 2021, ISBN 978-3-030-86981-6 p 57–72
- Keip P., Ferre S., Gutierrez A., Huchard M., Silvie S., Martin P., 2020. Practical Comparison of FCA Extensions to Model Indeterminate Value of Ternary Data. The 15th International Conference on Concept Lattices and Their Applications (CLA 2020). Tallinn, Estonia, 29-1/06-07/2020, p 197–208
- Keip P., Gutierrez A., Huchard M., Le Ber F., Sarter S., Silvie P., Martin P., 2019. Effects of Input Data Formalisation in Relational Concept Analysis for a Data Model with a Ternary Relation. The 15th International Conference on Formal Concept Analysis (ICFCA 2019), June 25 – 28 Frankfurt. Lecture Notes in Computer Science 11511, Springer 2019, p 191–207
- Bazin A., Carbonnel J., Huchard M., Kahn G., Keip P., Ouzerdine A., 2019. On-demand Relational Concept Analysis. The 15th International Conference on Formal Concept Analysis (ICFCA 2019), June 25 – 28 Frankfurt . Lecture Notes in Computer Science 11511, Springer 2019, p 155–172
- Keip P., Ouzerdine A., Huchard M., Silvie P., Martin P., 2019. Navigation conceptuelle dans une base de connaissances sur l'usage des plantes en santé animale et végétale. COnférence en Recherche d'Informations et Applications (CORIA 2019), Lyon, 25-29/03/19, 14p.

- Mahrach L., Gutierrez A., Huchard M., Keip P., Silvie P., Martin P., 2020. Extraction of association implication rules from knowledge on plants with pesticidal and antibiotic effect classified by FCA within the One-Health initiative. Poster aux Journées Ouvertes de Biologie, Informatique et Mathématique (JOBIM 2020). Montpellier, 30-3/06-07/2020

- Martin P., Keip P., Gutierrez A., Huchard M., Ilboudo Z., Sarter S., Tagne A., Silvie P., 2019. The Knomana knowledge base - A tool to promote exchange of knowledge and identify local plants for addressing sanitary problems in EOA. In : Book of abstracts - The 5th West African Organic Conference. Organic agriculture : Life for all. West Africa Organic Network. Accra : West Africa Organic Network, p. 19. West African Organic Conference (WAOC 2019). 5, Accra, Ghana, 12 Novembre 2019/15 Novembre 2019.

Les données des différentes expérimentations ont été déposées sur le dataverse du CIRAD. Le lecteur trouvera les références de ces dépôts ci-après :

- Martin P., Gutierrez A., Marnotte P., Huchard M., Keip P., Mahrach L., Silvie P., 2021. Dataset on Noctuidae species used to evaluate the Separate concerns in Conceptual Analysis: Application to a Life Sciences Knowledge Base. CIRAD Dataverse, V1, https://doi.org/10.18167/DVN1/HTFE8T

- Martin P., Gutierrez A., Huchard M., Keip P., Silvie P., 2020. Dataset of Knomana used to conduct the evaluation of RCA on large volumes of complex relational data. CIRAD Dataverse, V1, https://doi.org/10.18167/DVN1/5AASZE

- Martin P., Ferré S., Gutierrez A., Huchard M., Keip P., Silvie P., 2020. Dataset on Spodoptera used to conduct the practical comparison of FCA extensions to Model Indeterminate Value of Ternary Data. CIRAD Dataverse, V1, https://doi.org/10.18167/DVN1/VNCZYA

I

Construction et formulation de la problématique

La section introductive a présenté plusieurs SBC développés en agriculture, indiqué le contexte général du travail réalisé et son positionnement. Le SBC Knomana a été choisi, d'une part, pour son domaine d'application, i.e. l'agroécologie et l'approche One Health et, d'autre part, pour son état de prototype logiciel dont l'ambition est générique, i.e. permettre l'exploration des bases de connaissances qualitatives. En introduction, il est également indiqué que la méthode retenue pour explorer la base de connaissances est l'Analyse de Concepts Formels.

Dans ce chapitre, la section 1.1 présente de façon plus détaillée la base de connaissances du SBC Knomana. La section 1.2 présente les motivations ayant guidé le choix de l'Analyse de Concepts Formels. La section 1.3 présente la proposition du prototype d'architecture logicielle, qui a été nécessaire en préalable à la conduite du travail présenté dans ce mémoire. Afin de dégager la problématique centrale de la thèse, dont l'énoncé est présenté en section 1.4, un traitement parallèle de questions connexes concernant différents aspects du SBC, a été effectué. Il a été réalisé grâce au co-encadrement de plusieurs étudiants.

1.1 La base de connaissances Knomana

Knomana [Silvie et al., 2021], terme qui est la fracto-composition des mots « Knowledge » et « management », résulte d'un travail de capitalisation des connaissances initié en 2015 par Pierre Martin, informaticien au Cirad, et Pierre Silvie, entomologiste à l'IRD. La méthode de capitalisation adoptée a consisté à choisir un thème d'intérêt (par exemple la protection d'une culture agricole en utilisant un extrait d'une espèce végétale), puis à identifier un modèle d'énoncé de la connaissance ciblée, formalisé sous forme d'une ontologie, pour enfin recenser, dans la littérature (publication scientifique, mémoire d'étudiant, rapport technique, etc.) toutes les connaissances respectant ce modèle. Enfin,

des dictionnaires sont complétés de façon à standardiser les données utilisées pour décrire les connaissances, dans l'objectif de disposer de connaissances comparables voire composables. Les références bibliographiques d'où sont extraites les connaissances sont stockées dans une base bibliographique au format Zotero [Puckett, 2011].

Dans Knomana, les jeux de connaissances sont stockés dans un fichier tableur, où chaque onglet comporte un jeu de connaissances relatif à un thème d'intérêt. Le nom de l'onglet est celui du jeu de connaissances. Chronologiquement, le premier jeu de connaissances, appelé RAP (pour Ravageurs des *Poaceae*), rassemblait les chaînes trophiques des espèces de *Lépidoptères* foreurs de tiges ou d'épis (i.e. des bioagresseurs) de plantes d'Afrique appartenant aux familles *Poaceae*, *Cyperaceae* et *Typhaceae*. L'ontologie associée à ce jeu de connaissances connecte la plante hôte, le bioagresseur de la plante hôte, l'ennemi naturel du bioagresseur, le territoire où a été observée cette relation trophique (= connaissance), et enfin la référence bibliographique.

Avec RAP, l'objectif était de permettre d'identifier, par exemple dans un bas-fond rizicole au Bénin, les plantes locales non-cultivées (e.g. les mauvaises herbes) susceptibles d'héberger les bioagresseurs des cultures agricoles lorsqu'une culture (le riz, le sorgho ou le maïs, par exemple) était absente. L'intention était alors de gérer ces espaces non-cultivés, par exemple en coupant les plantes, de façon à limiter le développement de ces bioagresseurs au lieu de pulvériser des pesticides sur la culture agricole. L'intérêt de recenser les ennemis naturels était d'identifier les autres espèces de bioagresseurs localement présents qui leur servaient de lieu de reproduction de façon à favoriser la lutte biologique par conservation [Boucher et al., 2015]. Par la suite, l'application de ce travail au Sénégal, dans le cadre du développement d'un système d'aide à la décision [Martin et al., 2020], a mis en évidence la présence de certains bioagresseurs de la canne à sucre, culture majoritaire dans la zone d'étude, sur d'autres cultures voisines (e.g. riz, maïs) ainsi que sur une espèce de mauvaise herbe en très forte expansion dans les environs.

Les jeux de connaissances construits par la suite, relatifs à l'usage des plantes, ont été initiés du fait de leur importance d'usage, dans des domaines très variés (e.g. la médecine traditionnelle, la teinture naturelle, la pêche, l'alimentation humaine, etc.), en Afrique, en Amérique Latine et en Asie. Replacé dans le cadre du développement de l'agroécologie, l'intention est de favoriser le développement de ces plantes, cultivées ou non, utilisées dans ces régions.

En 2017 a été initié le jeu de connaissances PPAf (pour Plantes Pesticides d'Afrique). Il rassemble les connaissances sur les usages des plantes utilisées en agriculture comme alternative aux pesticides chimiques de synthèse. En 2018, l'objectif initial de ce jeu a été élargi dans le cadre d'un projet impliquant la collaboration de diverses entités : l'Université Joseph Ki-Zerbo (Burkina Faso), l'Institut de Recherche Agricole pour le développement (IRAD) du Cameroun, l'UMR ISEM et le LIRMM (`https://ur-aida.cirad.fr/nos-recherches/projets-et-expertises/knomana`). Ce faisant, les usages de plantes ayant un effet pesticide ou antibiotique, qu'ils concernent l'être humain (santé humaine et santé publique) ou les cultures agricoles animales (santé animale) et végétales, ont été recensés. La santé de l'environnement a été abordée à travers les connaissances liées aux effets non-intentionnels éventuellement rapportés sur les ennemis naturels des ravageurs.

Les sous-sections suivantes présentent un exemple de jeu de connaissances, PPAf en l'occurrence, et les dictionnaires de la base de connaissances.

1.1.1 Exemple de jeu de connaissances caractéristique - PPAf

Parmi les jeux de connaissances de Knomana, PPAf est celui qui comporte le plus grand nombre de connaissances, une connaissance correspondant à un usage de plante. C'est également celui dont les connaissances sont décrites par le plus grand nombre de types de données (70). En Octobre 2020, PPAf comportait 44 360 connaissances extraites de 342 documents de natures diverses (publication dans une revue scientifique, proceeding, bulletin académique, etc.), rédigés en français ou en anglais, et parus entre 1957 et 2020 [Silvie et al., 2021]. La saisie est toujours en cours.

Plusieurs versions de l'ontologie de PPAf ont été produites et implémentées avec l'éditeur d'ontologie Protégé. La figure 1.1 présente une première version de l'ontologie. Celle-ci indique, au moyen des relations Is-part-Of (flèche vertes), qu'un document rassemble des connaissances. Chaque connaissance est constituée d'un système à protéger (e.g. culture agricole, être humain), d'un bioagresseur et d'un biopesticide (i.e. extrait végétal). Les flèches noires indiquent que le biopesticide protège le système à protéger en agissant contre le bioagresseur de ce système. Une seconde version de l'ontologie de PPAf, également produite en 2018, est présentée en figure 1.2. Cette version diffère de la première par la matérialisation du système de protection, extrait de la connaissance, qui comporte un système à protéger, un bioagresseur et un biopesticide. Ce faisant, le document duquel est extrait la connaissance fait partie de la connaissance sur la protection, laquelle est une partie du système de protection. Par construction, le système de protection inclut les relations entre les trois constituants, i.e. attaque, protège et lutte.

La représentation de la connaissance diffère entre ces deux versions. Dans la première version, c'est le document qui est le point d'entrée de la description de la connaissance, alors que dans la seconde version, c'est le système de protection. Concernant le système de protection, deux modes de représentation d'une relation entre 3 éléments (relation ternaire) sont présentés. Dans la première version, les trois entités (prétendument constitutives du système de protection) sont connectées deux à deux par une relation sémantique binaire, alors que dans la seconde version, la relation ternaire est matérialisée par le concept « Système de protection » relié à chaque élément qui le constitue. Enfin, l'attribut « qualité » associé à la connaissance porte une sémantique différente dans les deux versions. En effet, dans la première version, la qualité peut concerner autant le contenu de la connaissance que la relation ternaire. Dans la seconde version, l'ambiguïté est levée : la qualité concerne exclusivement la qualité de la connaissance extraite du document, et non la qualité du système de protection à l'exemple de sa performance.

Dans l'onglet PPAf du fichier de tableur, une connaissance est décrite sur une ligne où chaque cellule comporte une donnée élémentaire descriptive de la connaissance. Le lien entre l'ontologie et le tableur s'effectue via le type de données élémentaire (i.e. la colonne) qui correspond à un attribut de l'ontologie. Dans les figures 1.1 et 1.2, pour des questions de présentation, certains types de données ont été regroupés, à l'exemple du pays où la plante est prélevée, de l'entité géographique (département ou région) et de la localité qui

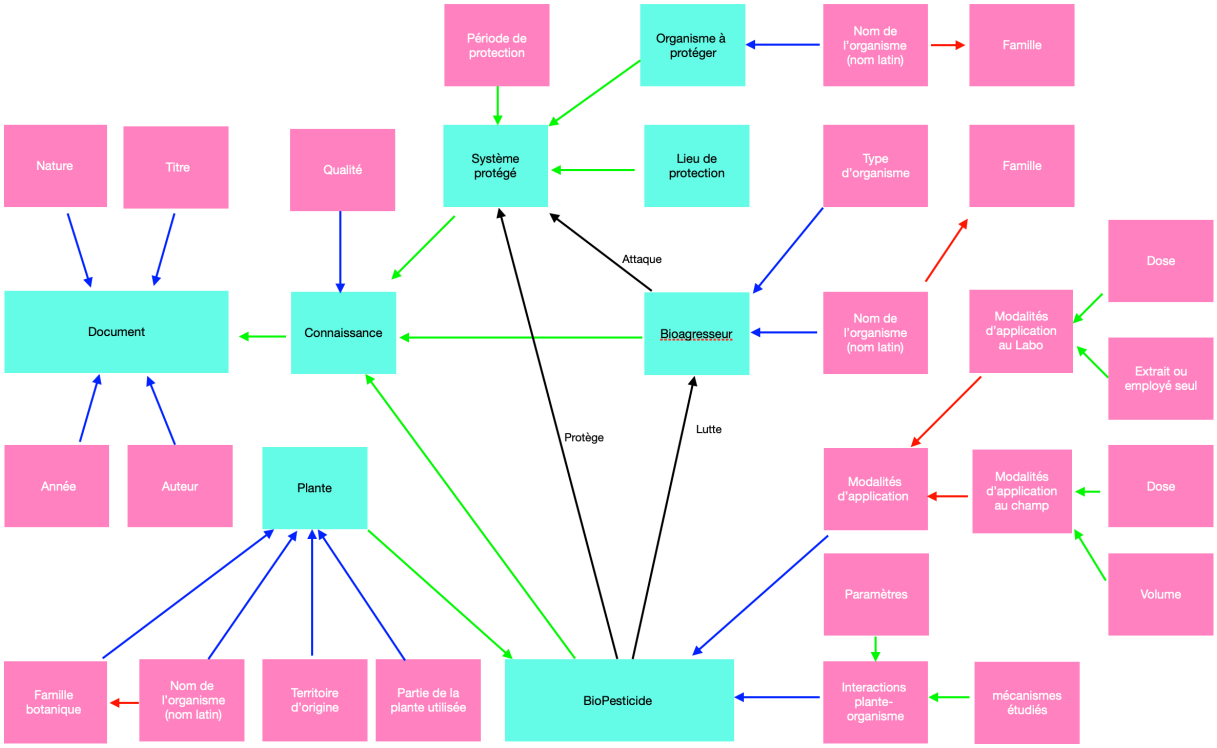


FIGURE 1.1 – *Première version de l'ontologie pour le jeu de connaissances PPAf, adaptée de [Marone, 2018]. Dans cette figure, les concepts et les attributs sont respectivement représentés par des rectangles verts et roses. Les flèches de couleur rouge, verte, et bleue représentent respectivement la relation Is-a, Is-Part-of et Is-Attribute-of. Une flèche de couleur noire représente une relation sémantique différente des 3 précédemment citées, et dont la signification est portée sur la flèche.*

sont regroupés au sein de l'attribut « Territoire d'origine ».

Dans cet onglet, les concepts sont indiqués sur la première ligne du tableau. Les attributs associés à chaque concept sont indiqués, dans des colonnes juxtaposées, sur la seconde ligne du tableau. Au-dessous de ces 2 lignes, chaque ligne comporte une connaissance. Dans la mesure où le système de gestion des données est inexistant puisque c'est un tableur, par convention, seul l'ajout de connaissances est permis en fin de tableau ; aucune manipulation des colonnes ou des lignes n'est autorisée.

Ce tableau comporte les données brutes extraites des documents. La description du document original, comportant la connaissance, est effectuée au moyen de plusieurs attributs, i.e. Auteur, Année, Titre du document, Revue, Volume, Pagination, Nature du document (e.g. livre, revue, etc.), Identifiant numérique d'objet (DOI). Dans la mesure où le document d'où est extraite la connaissance peut être une revue de la littérature, la référence à cette revue est également stockée.

Le format de la donnée présente dans chaque case varie selon l'attribut et peut présenter des ambiguïtés. Concernant le nom de la plante, bien que sa formulation scientifique soit préférée [1], on peut trouver dans la littérature consultée la forme vernaculaire

1. Nom latin binominal respectant le référentiel taxonomique des espèces (e.g. *Zea mays L.* où les

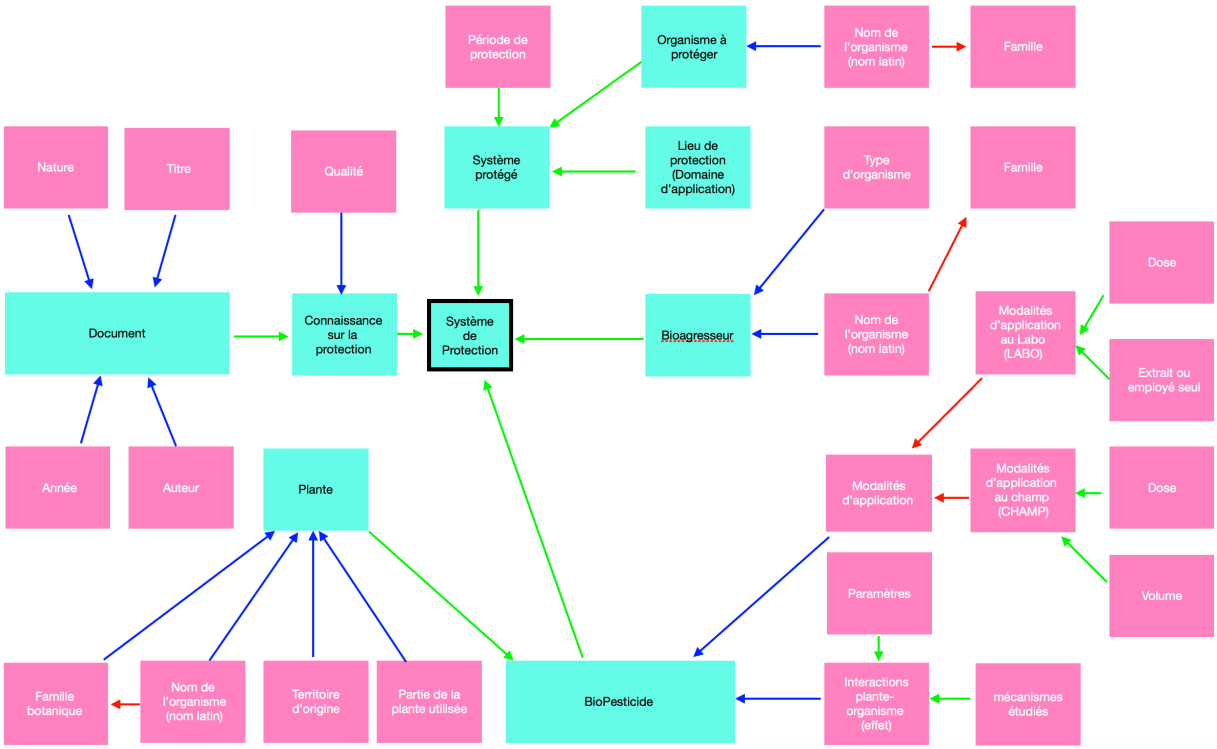


FIGURE 1.2 – *Seconde version de l'ontologie pour le jeu de connaissances PPAf, adaptée de [Marone, 2018]. Dans cette figure, les concepts et les attributs sont respectivement représentés par des rectangles verts et roses. Les flèches de couleur rouge, verte, et bleue représentent respectivement la relation Is-a, Is-Part-of et Is-Attribute-of.*

(i.e. maïs). Toutefois un nom vernaculaire peut faire référence à différentes espèces botaniques. Ainsi, le nom vernaculaire « Thym » fait référence aux espèces (plus de 300) du genre *Thymus*. Le « piment » fait référence aux fruits produits par cinq espèces du genre *Capsicum*. Le « poivre » fait référence à tous les fruits produits par les espèces de la famille *Piperaceae*, qui est constituée de cinq genres et de 3 600 espèces. À l'inverse, un nom scientifique peut disposer de noms vernaculaires différents selon les lieux ou les populations, à l'exemple d'une variété de *Zea mays L.* qui est appelée maïs doux en France et Blé d'Inde au Canada.

La nomenclature scientifique peut également présenter des sources d'ambiguïté. En particulier, deux termes sont utilisés pour désigner des espèces indéterminées ou un groupe d'espèces du même genre. Il s'agit des abréviations « sp. » et « spp. » L'abréviation sp. est utilisée pour mentionner une espèce indéterminée d'un genre. Par exemple, « *Capsicum* sp. » désigne une espèce du genre *Capsicum* sans en préciser le nom. « *Capsicum* spp. » fait référence à au moins deux espèces du genre *Capsicum* sans qu'il n'en soit précisé les noms. L'utilisation de ces abréviations entraîne donc l'indétermination sur la connaissance.

Pour la partie de la plante utilisée, un même objet peut être énoncé en diverses langues, au singulier ou pluriel, voire comporter des erreurs typographiques, provenant de l'article initial ou de la saisie manuelle. Ainsi, l'objet feuille est désigné sous les formes « feuille », « feuilles », « leaf », « leaves », « fzuille », « feiulle », « feulle », « feuillle », etc. Il ar-

termes *Zea*, *mays* et L. correspondent respectivement au genre, au nom d'espèce et à l'auteur

rive que plusieurs objets soient mentionnés et que le séparateur utilisé puisse induire un questionnement. Par exemple « leaf and seed » indique que l'on utilise les deux en même temps, tandis que « leaf/seed » ou « leaf, seed » laisse supposer que l'on peut utiliser l'un, l'autre ou les deux.

Pour les composés chimiques majeurs, ils sont séparés par des virgules, e.g. « Limonene, citronellal ». Dans certains cas, leur taux de présence est indiqué via un pourcentage, e.g. « Limonene oxide (38.99%), Verbenone (6.06%), Linalool (2.69%) ». Enfin, concernant l'attribut mécanisme étudié de l'interaction plante-organisme, on peut trouver un terme, e.g. « toxicité », son dual, e.g. « répulsion » et la combinaison des deux, i.e. « toxicité/répulsion ».

Du fait du grand nombre d'attributs identifiés pour décrire une connaissance, i.e. 70, le tableau présente une majorité de cellules non remplies. Cette situation était prévue par construction. Par exemple, des attributs étaient identifiés pour décrire les essais au laboratoire et d'autres pour ceux réalisés au champ. Dans la mesure où un travail de recherche publié ne couvre que très rarement les deux zones d'essais (notamment du fait des méthodes d'évaluation qui diffèrent), seul un de ces deux essais est généralement renseigné pour chaque connaissance. D'autre part, les revues de la littérature ne couvrent généralement qu'un aspect de l'usage des plantes, à l'exemple de la diversité d'espèces de plantes utilisées pour lutter contre un bioagresseur (e.g. [Boeke et al., 2004]) ou d'un extrait de plante utilisé sur différentes espèces de légumes stockés pour lutter contre plusieurs bioagresseurs (e.g. [Koona and Dorn, 2005]). *De facto*, il est très rare que la majorité des attributs soient renseignés. La question qui se pose est de savoir s'il est possible de renseigner certaines cellules vides pour compléter le jeu de connaissances afin de l'explorer, en utilisant par exemple les données présentes dans le jeu ou dans d'autres bases de données.

1.1.2 Les dictionnaires

Dans Knomana, seules sont saisies les données brutes. Pour disposer de connaissances comparables voire composables, la solution, adoptée dans les bases de données par exemple, a consisté à standardiser les données via la mise en place de dictionnaires de données [IBM, 1993]. Leur objectif est d'établir une correspondance entre chaque valeur de donnée brute et sa valeur standardisée.

Chaque dictionnaire est implémenté dans un onglet du fichier tableur et a été constitué pour chaque attribut de chaque jeu de données. Chacun comporte les valeurs de texte brut, leur équivalent standardisé ainsi qu'un commentaire. Pour certains dictionnaires, des classes ont été constituées par regroupements successifs de valeurs afin de répondre à des objectifs d'analyse. C'est le cas par exemple de l'attribut « Lieu de Protection », dont les valeurs brutes sont par exemple « animal d'élevage (poisson) », « Fromage traditionnel », « denrées stockées », « semences », « Protection au champ ». Les regroupements successifs des 38 valeurs brutes, via des classements intermédiaires, ont donné lieu à 5 classes principales, i.e. santé animale, santé végétale, santé humaine, santé publique et santé environnementale.

Trois dictionnaires généraux, i.e. utilisés par tous les jeux de données, ont été développés. Il s'agit du dictionnaire des espèces, de celui de la littérature grise et de celui des territoires.

Le dictionnaire des espèces rassemble toutes les valeurs désignant un organisme (vivant). Dans le cas de PPAf, ce dictionnaire comporte, pour valeur d'entrée, le nom des plantes utilisées pour produire le biopesticide, le nom des bioagresseurs et le nom des organismes protégés. En sortie, le dictionnaire comporte 6 niveaux taxonomiques, i.e. le règne, la classe, l'ordre, la famille, l'espèce et la sous-espèce. L'auteur du taxon le plus précis dans la description taxonomique, e.g. l'espèce ou la sous-espèce, est également indiquée. Des niveaux taxonomiques intermédiaires existent, à l'exemple de l'embranchement. Les 6 niveaux retenus correspondent à des besoins d'analyse et résultent d'échanges avec les chercheurs partenaires (e.g. botanistes). La désignation d'une espèce est effectuée au moyen de son nom scientifique (i.e. nom latin binominal).

La standardisation du nom des espèces est réalisée d'après les référentiels taxonomiques PlantsOfTheWorld.org pour les plantes, FishBase.org pour les poissons, et CatalogueOfLife.org pour les autres espèces. L'alignement de chaque espèce est vérifié manuellement. Un prototype d'alignement automatique, via l'utilisation d'un web service proposé par CatalogueOfLife.org, a été initié en 2019 dans le cadre d'un stage de Master 1 [Mahrach, 2019] que j'ai co-encadré. Ce prototype utilise, dans la requête, le nom de l'espèce et de la famille pour s'assurer de leur validité. Dans le cas où ces deux noms ne sont pas validés, le prototype recherche d'abord via le genre de la famille, puis uniquement la famille, dans la liste des noms d'espèces fournis par le service web, celui dont la distance de Levenstein [Levenshtein, 1966] est la plus faible. Lorsqu'aucun résultat satisfaisant n'est trouvé (i.e. l'erreur obtenue est supérieure au seuil permis par l'utilisateur), l'hypothèse est que le nom de famille comporte également des erreurs typographiques. Pour tester cette hypothèse, le prototype recherche alors la présence du genre de l'espèce dans le dictionnaire, pour ensuite corriger le nom de la famille et réévaluer le nom de l'espèce. Un second prototype a été développé par 4 étudiants dans le cadre d'un Travail d'Etude et de Recherche (TER) de Master 1 de l'Université de Montpellier [Cherfaoui et al., 2020], que j'ai également co-encadrés. Ce prototype utilise le service web Taxonomic Name Resolution Service (TNRS), qui implémente l'algorithme Taxamatch [Rees, 2014] développé spécialement pour traiter les noms d'espèce de la taxonomie.

Ces prototypes résolvent une majorité des problèmes d'alignement rencontrés mais pas celui des synonymes. En effet, la classification taxonomique des espèces évolue en lien avec les révisions menées par les taxonomistes. Une entité taxonomique (espèce) peut par exemple être considérée comme une sous-espèce après révision, à l'exemple de *Catantops spissus* Walker qui, d'après CatalogueOfLife.org, est répertoriée comme la sous-espèce *Oxycatantops spissus spissus* (Walker, F., 1870) de l'espèce *Oxycatantops spissus* (Walker, F., 1870). On peut noter au passage le changement du nom du genre *Catantops* en *Oxycatantops* indiquant une possible revue de la classification. Pour information, cette espèce est de la famille *Acrididae*, une famille d'insectes dont font partie les criquets.
D'autre cas de synonymie sont plus complexes à gérer, notamment lorsque l'auteur de l'espèce à modifier n'est pas indiqué dans la littérature consultée. Par exemple, l'espèce de plante *Aloe vera* sans indication du nom de l'auteur peut faire référence à *Aloe vera* (L.) Burm.f. ou bien à *Aloe succotrina* Weston suite à la reclassification de l'espèce *Aloe*

vera Mill. Dans certains cas, le choix vers l'un ou les autres peut être effectué d'après la localisation géographique où l'espèce a été recensée, sous réserve que les autres dénominations ne soient pas également présentes. Enfin, une attention particulière doit être portée sur les référentiels taxonomiques. Ainsi, alors que le référentiel pour les plantes était ThePlantList.org pendant plusieurs années, c'est PlantsOfTheWorld.org qui est désormais considéré (par les botanistes du Cirad notamment) du fait que le premier référentiel n'est plus maintenu.
Pour autant, les classifications des référentiels peuvent différer entre elles. Par exemple, le nom d'espèce de la plante *Hyptis suaveolens* (L.) Poit. est reconnu par ThePlantList.org mais considéré comme un synonyme de *Mesosphaerum suaveolens* (L.) Kuntze par PlantsOfTheWorld.org. Dans ce cas, c'est le nom du genre de la plante qui change. Ces différences peuvent avoir des répercussions non négligeables lors de la navigation/exploration des connaissances, notamment lorsque les abréviations sp. ou spp. sont utilisées pour indiquer un nom d'espèce dans la formulation des connaissances. Par exemple, si le nom de l'espèce donné dans la description de la connaissance est *Hyptis* sp., alors se pose la question de la prise en compte de *Mesosphaerum suaveolens* (L.) dans la mesure où le genre n'est pas le même d'après ce référentiel.

En octobre 2020, les règnes présents dans ce dictionnaire étaient Animalia, Bacteria, Chromista, Ciliophora, Fungi, Plantae, et Protozoa. Dans la mesure où la classification des virus est similaire à celle des organismes cellulaires, deux règnes ont été ajoutés dans cette liste, i.e. Heunggongvirae et Orthornavirae, pour les espèces de virus de poissons, e.g. *Salmonid herpervirus* et *Salmonid novirhabdovirus* respectivement. Le dictionnaire comportait 7 706 désignations d'organismes (i.e. données brutes). Après alignement avec les référentiels taxonomiques, ces désignations correspondent à 4 667 espèces d'organismes, soit 3 040 formulations supplémentaires de certaines des 4 667 espèces (synonymes, différentes orthographes du même nom).

Pour la littérature, le dictionnaire bibliographique comporte les octuplés de valeurs descriptives d'une référence bibliographique (auteur, année, etc.) et établit la correspondance avec les données de la base bibliographique au format Zotero.

Enfin, le dictionnaire des territoires comporte les valeurs brutes et leur équivalent standardisé recensé dans la base de données géographique GeoNames (http://www.geonames.org). Chaque donnée est complétée par le continent et la partie du continent (e.g. Afrique de l'Est, Afrique du Nord).

1.2 Motivations à utiliser l'Analyse de Concepts Formels pour explorer Knomana

Dans Knomana, les connaissances considérées relèvent de l'usage des plantes, avec un lien fort avec les santés végétale, animale, humaine et environnementale. Les jeux de connaissances inclus dans Knomana sont de taille modeste, e.g. quelques dizaines de milliers de connaissances décrites au mieux par quelques dizaines d'attributs dans le cas de PPAf. Ce nombre est très faible en regard de celui portant par exemple sur le génome, i.e. de quelques millions à quelques milliards de paires de base.

D'autre part, la majorité des données constitutives des connaissances sont de type symbolique, et, par conséquent, en minorité numérique. Dans le cas de PPAf, le nom de l'espèce de plante utilisée pour produire le biopesticide, à l'exemple de *Aloe vera* (L.), *Thymus vulgaris* L., etc., le nom de l'organisme ciblé, le nom de l'organisme du système à protéger, le territoire d'origine de la plante, la partie de la plante, les composés chimiques majeurs, etc. sont toutes des données symboliques multi-valuées.
Parmi les données numériques se trouvent l'année de publication, le numéro de volume et la pagination. La pagination comporte effectivement des chiffres, mais avec un format particulier à l'exemple de la fourchette pour la pagination, e.g. « 344-367 ». Des méthodes de conversion des données, à l'exemple du symbolique au numérique, sont présentées dans la littérature, comme dans [Poulos et al., 2005].
De plus, la connaissance est décrite selon une structure qu'il s'agit de prendre en compte au moment de l'analyse. Par exemple, le territoire, d'où est extraite la plante utilisée pour produire le biopesticide, n'est pas nécessairement le territoire où est appliqué le biopesticide. La plante peut par exemple avoir été importée puis vendue sur le marché où elle a été acquise. Il n'est ainsi pas possible d'associer systématiquement le territoire avec le bioagresseur et de conclure que le bioagresseur est présent sur ce territoire. Il en est de même avec la relation n-aire, à l'exemple de la relation ternaire (biopesticide, système protégé, organisme ciblé) de PPAf. En effet, un biopesticide α utilisé pour protéger un système β contre un bioagresseur χ, ne peut pas être considéré comme trois relations binaires (α, β), (β, χ), (α, χ) indépendantes car il est incohérent de considérer qu'un biopesticide protège un système sans connaître la cible du fait des trop nombreuses cibles pour lesquelles le biopesticide n'est pas fonctionnel. Par exemple un biopesticide qui permet de lutter contre une chenille ravageuse des fruits est très différent de celui utilisé pour protéger cette culture contre un oiseau. Cette structure n-aire est une connaissance qu'il s'agit de prendre en compte lors de la fouille des connaissances pour éviter des erreurs d'interprétation. De même, les données descriptives des connaissances ne sont pas toutes renseignées, et certaines sont indéterminées lorsque, par exemple, les abréviations sp. ou spp. sont utilisées.
Enfin, les jeux de connaissances sont non-équilibrés : peu de connaissances peuvent être récupérées sur un ravageur tandis que de nombreuses autres peuvent l'être, sur certaines espèces de plantes utilisées, en raison d'une orientation ponctuelle de la recherche de littérature en lien avec un problème urgent de ravageur à traiter, à l'exemple de la lutte contre l'espèce d'insecte exotique envahissant *Spodoptera frugiperda* Smith & Abbot (Lepidoptera : Noctuidae), ravageur du maïs.
Les connaissances de la base Knomana sont donc complexes à traiter. La question qui se pose est d'identifier la ou les méthodes susceptibles de permettre l'exploration de tels jeux de connaissances.

La littérature présente diverses méthodes pour fouiller un jeu de connaissances. Dans la mesure où la majorité des données est symbolique, les convertir toutes en données numériques présenterait un intérêt dans l'unique objectif de mobiliser une méthode qui apporterait une plus-value (e.g. fonctionnalité originale) inexistante dans les méthodes traitant des données symboliques.
Deux grandes approches s'intéressent à la résolution de problèmes, à l'analyse ou à l'apprentissage : l'approche sub-symbolique et l'approche symbolique [Lieberman, 2016].
L'approche sub-symbolique rassemble les méthodes statistiques et connexionnistes, à l'exemple du Deep learning [Schmidhuber, 2015], des random forests [Ho, 1995], et des

arbres de décision [Loh, 2014]. Ces méthodes permettent de traiter de gros volumes de données et sont suffisamment robustes pour taire les bruits dans les données. Un inconvénient du Deep learning et des random forests est la difficulté d'interprétation des calculs intermédiaires qui ont donné lieu au résultat final. Ce faisant, dans le cas où les résultats attendus ne sont pas satisfaisants, seule l'expérience permet d'émettre une hypothèse sur l'origine possible de cette différence. La méthode des arbres de décision donne accès, par construction, aux résultats intermédiaires dans la mesure où elle s'appuie sur la répartition statistique des similarités au sein du jeu de données. Néanmoins, cette répartition exige un jeu de données équilibré pour constituer l'arborescence décisionnelle ; ce qui n'est pas le cas avec une base de connaissances telle que Knomana.
L'approche symbolique rassemble les méthodes basées principalement sur la logique et l'explicitation des connaissances, à l'exemple de la programmation par contraintes [Rossi et al., 2006], des règles d'association [Piatetsky-Shapiro, 1991] et de l'Analyse de Concepts Formels. Ces méthodes s'appliquent aux jeux de données de taille modeste non équilibrés tels que ceux de Knomana et les résultats obtenus peuvent être explicités.
La programmation par contraintes est adaptée à Knomana mais nécessite, en particulier, la reformulation des connaissances ou de la question sous forme de contraintes. L'Analyse de Concepts Formels semble adaptée à Knomana et à d'autres jeux de connaissances sur l'environnement, comme présenté par [Braud et al., 2021]. Son intérêt est qu'elle prend les connaissances sans parti pris, mais formalisées sous la forme d'une matrice binaire appelée Contexte Formel, pour en proposer une représentation globale, consistant à les classer les unes relativement aux autres. L'extraction de règles d'associations est un champ qui pourra également être exploré et qui reste proche de l'Analyse de Concepts Formels, mais qui n'a pas pour objet de construire des classifications.

Dit simplement, l'Analyse de Concepts Formels génère un treillis classant un ensemble d'objets décrits par leurs attributs, à l'exemple d'objets permettant à un être humain de s'asseoir, objets décrits à l'aide d'une taille, d'un confort et d'une couleur. Outre cette apparente simplicité, diverses extensions de cette méthode ont été développées pour inclure différents aspects de l'exploration dans des jeux de données en lien avec les sciences du vivant, à l'exemple de la temporalité (e.g. [Nica et al., 2016]), du flou (e.g. [Rosa et al., 2021]) ou des relations entre objets (e.g. [Keip et al., 2019a]). Une des extensions de l'Analyse de Concepts Formels permettant de prendre en compte des relations entre objets s'appelle l'Analyse de Concepts Relationnels [Hacene et al., 2013]. Enfin, de façon alternative au treillis, des algorithmes permettent également de représenter les connaissances d'un contexte formel sous forme de règles d'implications (règles d'associations exactes). Par exemple la règle « *Spodoptera frugiperda*, Bénin → maïs » indique que « les connaissances concernant *Spodoptera frugiperda* au Bénin portent sur la culture du maïs ». L'intérêt de ce formalisme est qu'il s'apparente à celui utilisé par les moteurs d'inférence pour raisonner sur les connaissances dans un objectif d'aide à la décision.

1.3 Proposition d'architecture logicielle pour Knomana

Transformer la base de connaissances Knomana en un système à base de connaissances implique de développer une application logicielle permettant d'explorer les connaissances. En préalable à la proposition d'une architecture logicielle et du langage de modélisation interne des données, il est nécessaire d'identifier les utilisations potentielles d'une telle application.

1.3.1 Exemples d'utilisation de Knomana

La base de connaissances Knomana a pour ambition première de doter les chercheurs impliqués dans la protection des cultures végétales d'un capital cognitif pour identifier des solutions alternatives à l'utilisation des pesticides. Par exemple, le jeu RAP, comportant les chaînes trophiques (ennemi naturel, bioagresseur, plante hôte), permet d'identifier les ennemis naturels de bioagresseurs, dont il s'agirait de renforcer l'activité biologique pour limiter le recours aux pesticides. Lorsque les ennemis naturels ne sont pas suffisamment efficaces, par exemple du fait du trop grand nombre de bioagresseurs, alors le jeu de connaissances PPAf permet d'identifier la ou les plantes locales à partir desquelles un biopesticide peut être préparé puis appliqué sur la culture agricole.

L'extension de PPAf, portant initialement sur les plantes à effet pesticide, aux plantes présentant une action antibiotique a conduit à élargir le spectre des utilisateurs de Knomana aux chercheurs impliqués dans la protection phytosanitaire des animaux terrestres (bovins, ovins, etc.) et aquatiques (poisson, crevette, etc.) et aux chercheurs impliqués en santé publique (e.g. lutte contre les vecteurs de maladies comme les moustiques) Dès lors qu'il s'agit de protéger l'être humain, à l'exemple des effets induits par l'emploi excessif des antibiotiques en aquaculture (cf. exemple donné dans le chapitre introduction), la collaboration avec les médecins devient incontournable car les plantes, identifiées pour remplacer les antibiotiques, tout comme celles employées en santé végétale, sont susceptibles de comporter des composés naturels nocifs pour l'être humain.

L'adoption de l'agroécologie comme modèle de production agricole conduit le producteur agricole à prendre en compte son environnement (e.g. la faune et la flore locale) dans le pilotage de son exploitation agricole. Dans la mesure où chaque parcelle cultivée est située dans un environnement spécifique, les chercheurs ne sont pas en capacité de se mobiliser pour répondre individuellement aux besoins de chacun et proposer, par exemple, la plante à effet pesticide qui convient. Ce faisant, l'application logicielle à développer doit également pouvoir être utilisée par les consultants, voire les producteurs agricoles.

Cette application doit permettre de prendre en compte, non seulement la problématique phytosanitaire, mais également les aspects néfastes de la plante, à l'exemple de la toxicité de la plante sur le technicien qui applique le biopesticide. Ainsi, pour répondre à une problématique de l'approche One Health, l'enjeu est de concilier les santés humaines et environnementales. Par exemple, dans [Mahrach et al., 2020] et [Mahrach et al., 2021], nous nous sommes posés la question de définir une démarche d'identification de plantes qui

respecte l'approche One Health, pour protéger une culture agricole contre un bioagresseur de la famille Noctuidae. La contrainte était que la plante de protection ne devait pas être utilisée en soin médical dans le cas où le produit de la culture agricole était consommé. En effet, l'hypothèse est que, de façon analogue à un antibiotique, la plante utilisée pour produire le biopesticide est susceptible de comporter des composés chimiques nocifs pour l'être humain et que des résidus de ce biopesticide peuvent se retrouver sur le produit consommé. Ce travail a été conduit dans le cadre d'un stage de Master 2 [Mahrach, 2020] que j'ai co-encadré. Une fois la méthode développée dans ce travail introduite dans l'application logicielle, celle-ci pourrait, selon l'utilisateur, présenter uniquement les plantes respectant la contrainte (i.e. ne pas être toxique pour le genre humain), voire celles qui ne la respectent pas tout en alertant l'utilisateur sur les risques encourus.

Pour répondre à ces exemples d'utilisation, l'application logicielle doit permettre d'explorer chaque jeu de connaissances. Par exemple, dans [Keip et al., 2019b], nous présentons une stratégie d'exploration du jeu PPAf permettant d'identifier une plante locale, en l'absence de connaissances locales. La solution a reposé sur la combinaison de plusieurs connaissances pour identifier dans un pays tiers une plante, présente localement, utilisée pour répondre à un problème phytosanitaire connexe, le même genre de bioagresseur en l'occurrence. Le co-encadrement d'étudiants dans le cadre de stage de Master 1, i.e. [Samb, 2019] et [Mbodj, 2020], puis d'un groupe d'étudiants dans le cadre d'un TER [Diboune et al., 2020], ont permis de dresser des premières pistes de présentation de données relatives à la relation ternaire (présentée en section 1.2).

L'application logicielle doit également permettre d'explorer la combinaison de jeux de connaissances. Par exemple, dans [Mahrach et al., 2020] et [Mahrach et al., 2021], résultant d'un travail conduit dans un stage de Master 2 [Mahrach, 2020] que j'ai co-encadré, nous combinons des connaissances sur l'usage des plantes en santé végétale (jeu PPAf) avec des connaissances sur leur usage en alimentation humaine (jeu Plantes Alimentaires), ainsi qu'en soin médical. Pour les usages en soin médical, ces connaissances ont été préalablement extraites de PPAf pour constituer un jeu de connaissances supplémentaire, mais présentées différemment de celles de PPAf. Comme le montre ce cas d'application, l'application logicielle ne peut donc pas être inféodée à un ensemble de jeux de connaissances, pour permettre d'en analyser une combinaison. L'obtention de cette combinaison nécessite en premier lieu la construction d'un modèle descriptif du jeu de données final à constituer, qui peut par exemple être formalisé sous forme d'une ontologie. Cette ontologie pourrait comporter des relations n-aires entre les données, ces données étant préalablement uniformisées au moyen des dictionnaires.

1.3.2 Architecture logicielle envisagée

L'architecture logicielle envisagée comporterait deux principaux composants interreliés (Figure 1.3) : le composant Knowledge Base comporterait la base de connaissances, et le composant Conceptual Harvester (ou moissonneur conceptuel) le système logiciel qui permettrait de fouiller la base de connaissances. L'intérêt d'une telle architecture, qui isole les données de leur traitement, est d'offrir la possibilité de traiter des bases de connaissances autres que Knomana, stockées localement ou distantes (sur le réseau Internet par

exemple), seules ou composées avec d'autres.

Le moissonneur conceptuel comporterait quatre sous-composants (Figure 1.3). Le sous-composant Query builder permettrait la formulation des requêtes par l'utilisateur. Le sous-composant Knowledge extractor permettrait d'extraire les connaissances du composant Knowledge Base en réponse à la requête élaborée par l'utilisateur via le composant Query builder. Le composant Reasoning Tool permettrait de compiler les connaissances et les structurer de façon à pouvoir y naviguer et les explorer au moyen du composant Knowledge viewer.

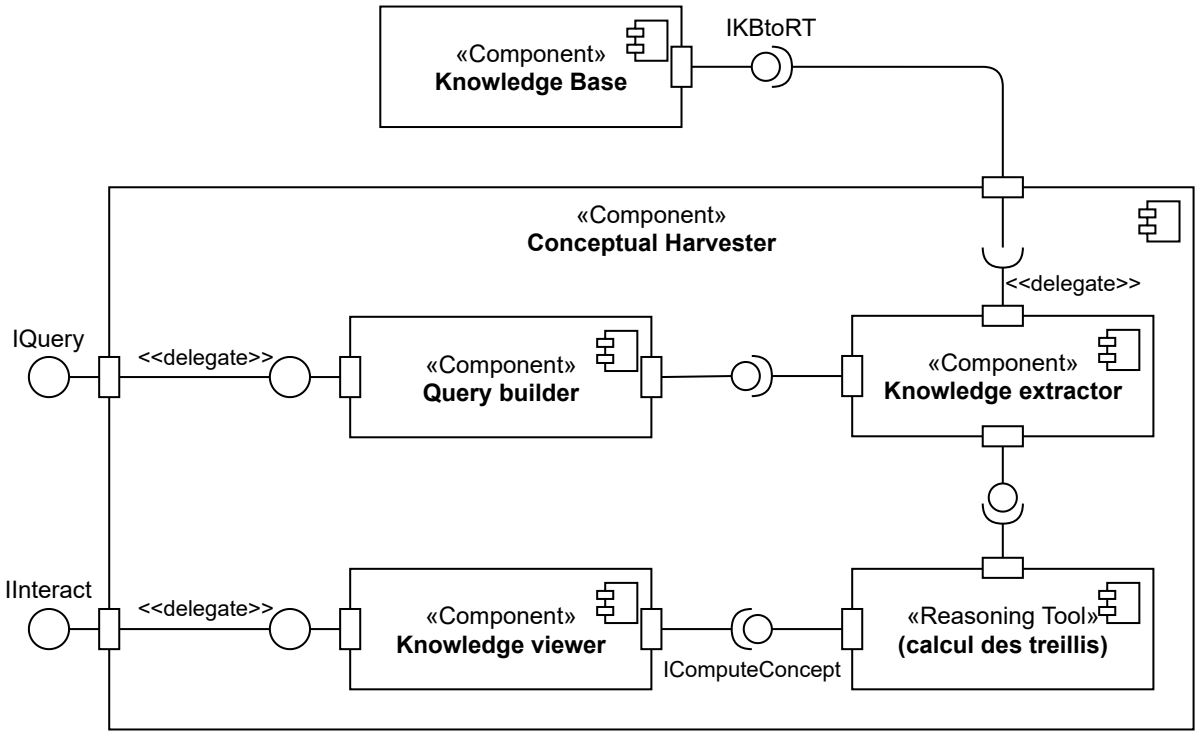


FIGURE 1.3 – *Diagramme de composants, au format UML, proposée pour supporter le système à base de connaissances Knomana. Adapté de [Keip et al., 2019b].*

Le composant Reasoning Tool pourrait, par exemple, comporter la suite logicielle Waikato environment for knowledge analysis [Garner, 1995] appelé plus généralement Weka (`http://www.cs.waikato.ac.nz/~ml/weka`). En effet, Weka comporte des algorithmes basés sur l'apprentissage automatique pour la fouille de données. Le Reasoning Tool pourrait, de même, comporter le logiciel RCAexplore [Dolques et al., 2019]. RCAexplore calcule les structures conceptuelles (treillis ou structures dérivées) au moyen de l'Analyse de Concepts Formels [Ganter and Wille, 1999] ou de l'Analyse de Concepts Relationnels [Hacene et al., 2013] en totalité, par étapes [Dolques et al., 2013b], ou à la volée à partir d'un groupe d'attributs ou d'objets d'intérêt [Bazin et al., 2019]. Dans la mesure où RCAexplore nécessite une adaptation logicielle du fait qu'il dispose d'une interface utilisateur et propose une visualisation non interactive des structures conceptuelles, l'alternative serait d'intégrer la librairie FCA4J (`https://www.lirmm.fr/fca4j/`) développée au LIRMM, implémentant des algorithmes similaires. En l'état actuel des développements, contrairement à RCAexplore, FCA4J permet de calculer des structures conceptuelles finales, et non par étape ou à la volée, ainsi que de formuler les connaissances sous forme de règles d'implication. Pour la visualisation avec le composant Knowledge viewer, une solu-

tion disponible est l'outil RCAviz (https://info-demo.lirmm.fr/rcaviz/), développé au LIRMM, qui permet de naviguer dans des structures conceptuelles construites avec l'Analyse de Concepts Relationnels.

Le diagramme de cas d'utilisation du composant Conceptual Harvester est présenté en Figure 1.4. Après avoir saisi une requête (Build query) via le composant Query builder, l'utilisateur interagirait avec les connaissances (interact) via le composant Knowledge viewer. Différentes formes d'interactions seraient permises. Dans le cas de la navigation (navigate) par exemple, l'utilisateur aurait la possibilité de changer le point de vue de la visualisation des connaissances affichées (change viewpoint), e.g. en passant d'un concept à un autre via les différents éléments affichés à l'écran, ou alors de se concentrer sur une connaissance ou un groupe de connaissances (focus on pieces of knowledge) pour en obtenir toutes les informations détaillées y compris celles qui n'étaient pas sélectionnées pour la navigation. L'utilisateur aurait également la possibilité d'effectuer des calculs (launch reasoning tool) que ce soit pour effectuer des calculs sur le jeu de connaissances affiché (compute globally) ou seulement pour une partie (compute locally). Cette distinction navigate vs. launch reasoning tool correspond respectivement aux fonctions explore et browse définies par [Munzner, 2014]. Ces cas d'utilisation sont génériques. Ils pourraient donc être adaptés (i.e. spécialisés) à chaque type d'utilisateur.

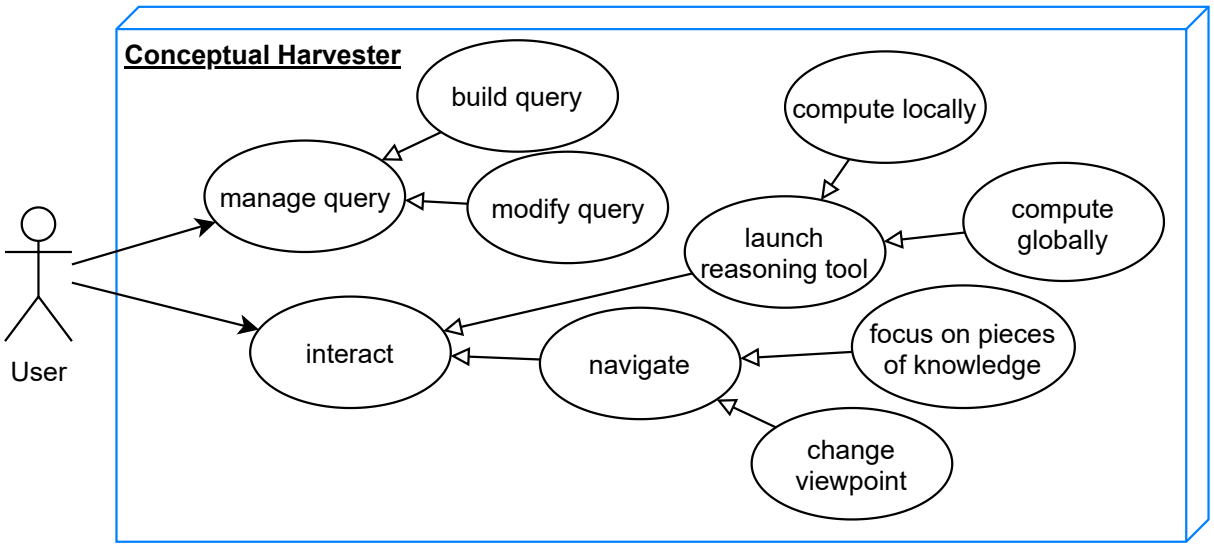


FIGURE 1.4 – *Diagramme de cas d'utilisation du composant Conceptual Harvester au format UML.*

1.3.3 Langage de modélisation interne des données

La littérature présente de nombreux langages de modélisation des données pour représenter l'information sous forme numérique. Parmi les langages célèbres produits par l'Intelligence Artificielle, on peut citer les frame-based languages [Bobrow and Winograd, 1977, Brachman and Schmolze, 1985], les réseaux sémantiques, graphes de connaissances et graphes conceptuels [Sowa, 1984, Sowa, 1991, Chein and Mugnier, 2009] et les logiques de description [Baader et al., 2017]. Plus récemment, l'initiative du Web sémantique a popularisé les formats RDF (Resource Description Framework), RDFS (Resource Des-

cription Framework Schema) [W3C, 2014] et OWL (Web Ontology Language) [Bechhofer et al., 2004] qui sont une formalisation simplifiée de ces langages, et permettent par exemple exclusivement l'expression des relations binaires. Les domaines des bases de données et du génie logiciel ont, de leur côté, également développé des langages de modélisation à l'exemple des modèles EER (Enhanced Entity-Relationship) [Teorey et al., 1986] et d'UML (Unified Modeling Language ; [Rumbaugh et al., 2004]). Dans la mesure où plusieurs auteurs utilisent UML pour modéliser les ontologies, e.g. [Wang and Chan, 2001, Kogut et al., 2002, Baclawski et al., 2002], formalisées par la suite avec le langage OWL [Vo and Hoang, 2020], nous avons choisi UML comme langage de modélisation des données pour l'application logicielle.

1.4 Problématique de la thèse

L'application logicielle, présentée dans la section précédente, a pour intention première d'offrir un cadre générique qui permettrait de fouiller toute base de connaissances sur l'environnement, et non pas exclusivement Knomana. La décomposition de l'architecture logicielle en composants élémentaires permet d'isoler les travaux de développement informatique, et de situer les problématiques abordées dans ce mémoire.

Le composant Reasoning Tool a pour fonction de compiler les connaissances et de les structurer. Autrement dit, son rôle consiste à préparer le jeu de connaissances, résultant ou non de la combinaison de plusieurs autres jeux, en perspective de son exploration par l'utilisateur final. C'est dans ce cadre que s'inscrivent les travaux conduits dans ce mémoire. Trois problématiques ont été identifiées. Leur degré d'application aux connaissances en agriculture est graduel, du plus général au cas particulier.

La première problématique concerne la conversion automatique d'un jeu de connaissances. Dans la mesure où la majorité des jeux de données en lien avec l'agriculture et l'environnement sont généralement stockés dans une base de données relationnelle, le choix du modèle de description du jeu de données est le diagramme de classes du langage UML. Dans le cas où ce diagramme comporte un ensemble de classes connectées, c'est l'extension Analyse de Concepts Relationnels qui est utilisée. Cela implique que le jeu de données doit être formaté en regard de cette extension. Ce formatage consiste à produire un ensemble de tables binaires interconnectées, appelée Famille de Contextes Relationnels, constituée de Contextes Formels (comportant les liens entre objets et attributs) et de Contextes Relationnels (comportant les liens entre objets).

La seconde problématique concerne l'étude des diverses modélisations possibles d'une relation n-aire. Nous avons précédemment montré que trois relations binaires ne sont pas équivalentes à une relation ternaire. Aussi, de façon usuelle, une relation ternaire est généralement convertie sous la forme d'une classe intermédiaire, reliée avec chacune des autres classes. Vis-à-vis de l'Analyse de Concepts Relationnels, chaque classe est convertie sous la forme d'un contexte formel, puis un treillis est construit pour chaque contexte formel. Or, l'étude des treillis est laborieuse pour un utilisateur, car il doit sans cesse passer de l'un à l'autre. La solution consiste donc à réduire le nombre de treillis, et donc le nombre de classes, en transformant la relation ternaire en trois relations binaires. La

problématique n°2 est par conséquent d'évaluer l'impact de diverses modélisations d'une relation n-aire sur la classification des connaissances.

La troisième problématique concerne la levée de l'indétermination des données. Dans Knomana, la désignation du nom de l'espèce d'un organisme comporte parfois l'abréviation spp. L'usage de cette abréviation indique que ce nom fait, *de facto* référence à au moins deux espèces du même genre. Le nombre et le nom des espèces d'organismes associées à ce nom est par conséquent indéterminé. Le jeu de connaissances comportant cette donnée indéterminée peut comporter plusieurs espèces déterminées de ce genre. Des hypothèses peuvent être émises pour lever cette indétermination. Par exemple, on peut considérer que spp. recouvre toutes les espèces du genre présentes dans le jeu, seulement une partie, ou aucune. La problématique n°3 adressée dans ce mémoire est d'identifier une modélisation de la levée de cette indétermination et d'en étudier l'impact sur la classification des connaissances.

Pour illustrer ces 3 problématiques, divers jeux de la base de connaissances Knomana ont été choisis. Ces jeux sont présentés au gré de leur utilisation.

II

Analyse de Concepts Formels et Relationnels

La section 1.2 a présenté les raisons qui ont conduit à utiliser l'Analyse de Concepts Formels (FCA) pour explorer la base de connaissances Knomana. Ce chapitre présente tout d'abord FCA (section 2.1) et les différentes structures conceptuelles qu'il est possible de produire avec FCA (section 2.2). Les sections suivantes présentent des extensions relationnelles de FCA, i.e. l'analyse de concepts relationnels (RCA) en section 2.3, l'analyse triadique de concepts (TCA) en section 2.4 et l'analyse conceptuelle de graphes de connaissances (GFCA) en section 2.5. La section 2.6 présente les différents types de transformation d'attributs multi-valués en attributs booléens, par des techniques de *scaling* (pour graduation) recensées dans la littérature. Puis nous concluons (section 2.7).

De façon à faciliter la compréhension des méthodes, les notions ne sont pas toutes mathématiquement formalisées dans ce chapitre. Seules les informations essentielles à la compréhension le sont. Pour plus d'informations, le lecteur est invité à se reporter aux articles cités.

2.1 Analyse de Concepts Formels (FCA)

L'analyse de concepts formels, souvent appelée l'analyse formelle de concepts [Ganter and Wille, 1999] est une méthode de classification basée sur les treillis de Galois. Elle utilise des données organisées sous forme tabulaire, que l'on appelle un contexte formel. Un contexte formel $\mathcal{K}$ met en relation un ensemble d'objets et un ensemble d'attributs. $\mathcal{K}$ est donc un 3-tuple (G, M, I) où G est l'ensemble des objets, M est l'ensemble des attributs et I est une relation binaire entre G et M, $I \subseteq G \times M$. La table 2.1 présente deux contextes formels qui servent d'exemples pour les sections 2.1, 2.2 et 2.3. La table 2.1a présente le contexte formel $\mathcal{K}_{Pest}$ décrivant des espèces d'agresseurs. Chaque ligne représente une espèce. Les trois premières colonnes indiquent le genre de ces espèces. Les trois colonnes

suivantes indiquent les pays dans lesquels ils sont présents. Les deux colonnes suivantes indiquent les organismes attaqués par ces agresseurs. Et enfin, la dernière colonne indique leur règne. La table 2.1b présente un contexte formel $\mathcal{K}_{Plant}$ décrivant une plante pour produire un biopesticide en utilisant certains de ses organes, l'état du produit cultivé (i.e. stocké pour être par exemple utilisé sous forme de semence ou utilisé pour l'alimentation humaine) et son genre.

$\mathcal{K}_{Pest}$	Callosobruchus	Rhipicephalus	Spodoptera	Burkina	Cameroun	Benin	goat	niebe	Animalia
CallosobruchusChinensis	×			×				×	×
CallosobruchusMaculatus	×				×			×	×
RhipicephalusLunulatus		×			×	×	×		×
RhipicephalusMicroplus		×			×	×	×		×
SpodopteraLittoralis			×		×	×			×

(a) *Contexte formel* $\mathcal{K}_{Pest}$

$\mathcal{K}_{Plant}$	partUsed:flower	partUsed:leaf	partUsed:root	partUsed:bark	storage	food	Hyptis	Mentha	Ocimum
HyptisSuaveolens					×		×		
HyptisSpicigera					×		×		
MenthaPulegium		×				×		×	
MenthaSpicata						×		×	
MenthaSuaveolens		×			×	×		×	
MenthaVirescens	×	×			×	×		×	
OcimumBasilicum	×	×	×	×	×	×			×
OcimumGratissimum	×	×	×	×	×	×			×

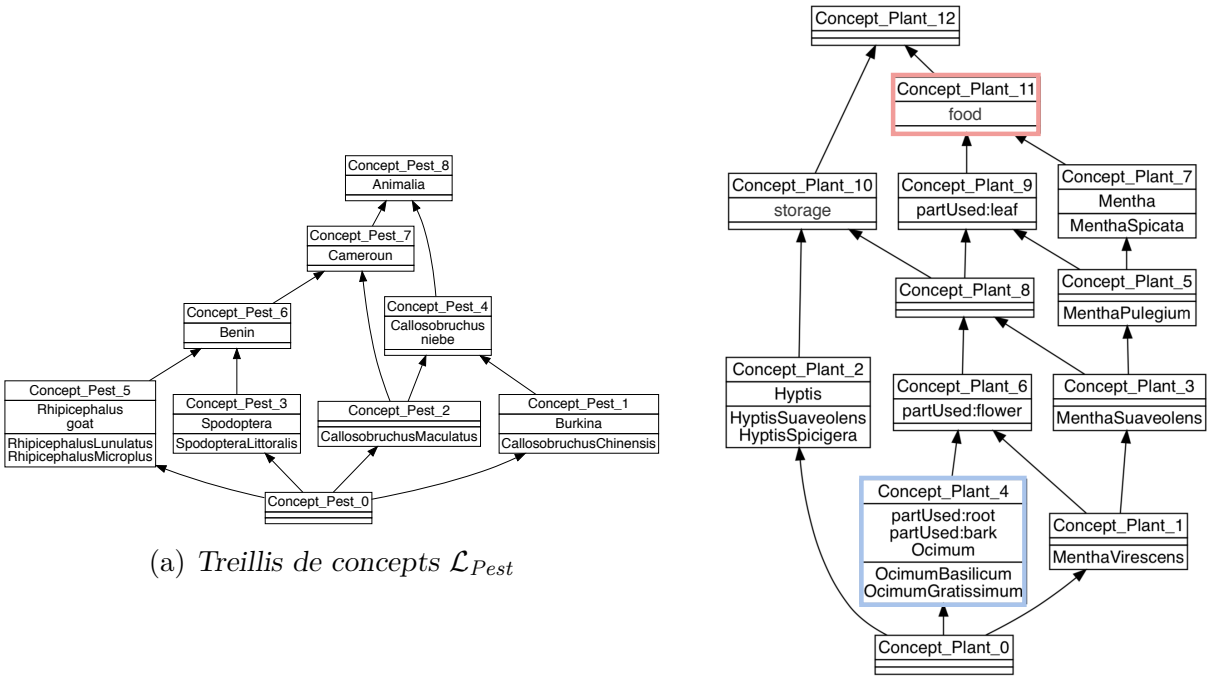
(b) *Contexte formel* $\mathcal{K}_{Plant}$

TABLE 2.1 – *Contextes formels* $\mathcal{K}_{Pest}$ *et* $\mathcal{K}_{Plant}$ *décrivant respectivement des agresseurs et des plantes à l'aide d'attributs. Les cellules rouges représentent le concept des plantes protégeant un organisme servant de nourriture. Les cellules bleues représentent le concept des plantes du genre Ocimum et toutes leurs caractéristiques.*

A partir d'un contexte formel, FCA forme des groupes d'objets et d'attributs, un groupe étant appelé concept formel, ou plus simplement un concept. Un concept $\mathcal{C} = (Extent(\mathcal{C}), Intent(\mathcal{C}))$ est un ensemble maximal d'objets partageant un ensemble maximal d'attributs. Par exemple, les cellules rouges de la Table 2.1b correspondent aux plantes protégeant un organisme pouvant être consommé (e.g. des tomates ou du maïs). Ces cellules peuvent donc être rassemblées pour donner lieu au concept $\mathcal{C}_{food}$. De la même façon, les cellules bleues correspondent au concept $\mathcal{C}_{Ocim}$ des plantes appartenant au genre *Ocimum*. L'ensemble des objets d'un concept forme son extension : $Extent(\mathcal{C}) = \{g \in G \mid \forall m \in Intent(\mathcal{C}), (g,m) \in I\}$. L'ensemble des attributs d'un concept forme son intension [1] : $Intent(\mathcal{C}) = \{m \in M \mid \forall g \in Extent(\mathcal{C}), (g,m) \in I\}$. Après avoir été identifiés, les concepts sont ensuite organisés hiérarchiquement en utilisant l'ordre de spécialisation $\preceq_{\mathcal{C}}$. Cet ordre les classe selon l'inclusion de leurs ensembles d'objets et d'attributs. Par exemple, $Intent(\mathcal{C}_{food}) \subseteq Intent(\mathcal{C}_{Ocim})$ et $Extent(\mathcal{C}_{Ocim}) \subseteq Extent(\mathcal{C}_{food})$ donc on obtient $\mathcal{C}_{Ocim} \preceq_{\mathcal{C}} \mathcal{C}_{food}$. $\mathcal{C}_{Ocim}$ est donc plus spécifique que $\mathcal{C}_{food}$.

L'ordre $\preceq_{\mathcal{C}}$ munit l'ensemble de tous les concepts d'une structure de treillis. On l'appelle un treillis de concepts et on le notera $\mathcal{L}$. Dans les représentations graphiques des treillis données dans ce document, les concepts les plus spécifiques sont placés en dessous des concepts qui leur sont plus généraux. La figure 2.1 montre les treillis $\mathcal{L}_{Pest}$ (figure 2.1a) et $\mathcal{L}_{Plant}$ (figure 2.1b) produits respectivement à partir des contextes formels $\mathcal{K}_{Pest}$ et $\mathcal{K}_{Plant}$. Les concepts rouge et bleu de la figure 2.1b (respectivement *Concept_Plant_11* et *Concept_Plant_4*) représentent les concepts $\mathcal{C}_{food}$ et $\mathcal{C}_{Ocim}$. $\mathcal{C}_{Ocim}$ étant plus spécifique que $\mathcal{C}_{food}$ d'après $\preceq_{\mathcal{C}}$, il se retrouve graphiquement en-dessous. On dit que $\mathcal{C}_{Ocim}$ est un sous-concept de $\mathcal{C}_{food}$ et inversement que $\mathcal{C}_{food}$ est un super-concept de $\mathcal{C}_{Ocim}$.

1. Pour cette orthographe particulière voir : https://fr.wikipedia.org/wiki/Intension_et_extension



(a) *Treillis de concepts* $\mathcal{L}_{Pest}$

(b) *Treillis de concepts* $\mathcal{L}_{Plant}$

FIGURE 2.1 – *Réduction transitive des treillis de concepts* $\mathcal{L}_{Pest}$ *(a) et* $\mathcal{L}_{Plant}$ *(b) produits par FCA en utilisant les contextes formels* $\mathcal{K}_{Plant}$ *et* $\mathcal{K}_{Pest}$ *(voir Table 2.1). Les concepts 11 (en rouge) et 4 (en bleu) correspondent aux concepts mis en évidence dans la table 2.1, respectivement* $\mathcal{C}_{food}$ *et* $\mathcal{C}_{Ocim}$.

La représentation graphique des treillis présentée en figure 2.1 est de plus simplifiée. Tout d'abord, elle ne montre que la réduction transitive de l'ordre de spécialisation. Ainsi, la relation entre les concepts 11 et 4 n'est pas directement explicitée car elle est implicitement présente grâce aux 4 relations entre les concepts 11, 9, 8, 6 et 4. Ensuite, les objets et les attributs ne sont pas inscrits dans chaque concept qui les possède mais seulement dans ceux qui les introduisent, concepts que l'on qualifie d'introducteurs. Un concept introducteur est un concept qui contient la première occurrence d'un attribut ou d'un objet, i.e. l'occurrence qui est située respectivement le plus haut ou le plus bas dans le treillis. Par exemple, le concept rouge est le concept introducteur de l'attribut *food* car il est le plus haut à posséder cet attribut. Un attribut introduit par un concept est présent dans tous les sous-concepts de celui-ci. De même, le concept bleu est le concept introducteur des objets *OcimumBasilicum* et *OcimumGratissimum* car il est le plus bas à posséder ces deux objets. Tout objet introduit par un concept est présent dans tous les super-concepts de celui-ci. Par ailleurs, un concept peut être à la fois introducteur d'objet et d'attribut, à l'exemple du concept bleu, ou n'être introducteur ni d'objet, ni d'attribut, comme le concept 8. Dans ce dernier cas, il s'agit d'un concept non-introducteur.

2.2 Les structures conceptuelles de FCA

Différentes structures conceptuelles peuvent être construites à partir d'un contexte formel. Cette section présente deux de ces structures conceptuelles : les AOC-posets (pour

« partially ordered sets of Attribute-Object Concepts ») [Dicky et al., 1995] et les treillis Iceberg [Stumme et al., 2002]. Pour faciliter la compréhension de ces deux structures, elles sont présentées en regard des treillis de concepts, même si partir des treillis de concepts n'est pas un procédé efficace pour les construire.

AOC-poset L'AOC-poset [Dolques et al., 2013a] également appelé « Galois Sub-Hierarchy » [Dicky et al., 1995] correspond à un treillis de concepts auquel les concepts non introducteurs ont été retirés. La figure 2.2 présente ces deux structures. Ayant été construites indépendamment l'une de l'autre par le logiciel RCAexplore, les numéros des concepts de même extension diffèrent entre les structures. Par exemple, le Concept *Concept_plant_11* du treillis de concepts (à gauche de la figure 2.2) correspond au concept *Concept_plant_9* de l'AOC-poset (à droite de la figure 2.2). Dans le treillis de concepts (à gauche), les concepts marqués d'une croix, i.e. les *Concept_Plant_i* avec $i \in \{12, 8, 0\}$, sont des concepts qui n'introduisent ni attribut, ni objet. Ils ne sont donc pas conservés dans l'AOC-poset correspondant. De même, les relations d'ordre relatives à ces concepts sont mises à jour. Des relations transitives comme celle qui reliait par exemple *Concept_Plant_6* à *Concept_Plant_10* sont conservées (en rouge sur la figure).

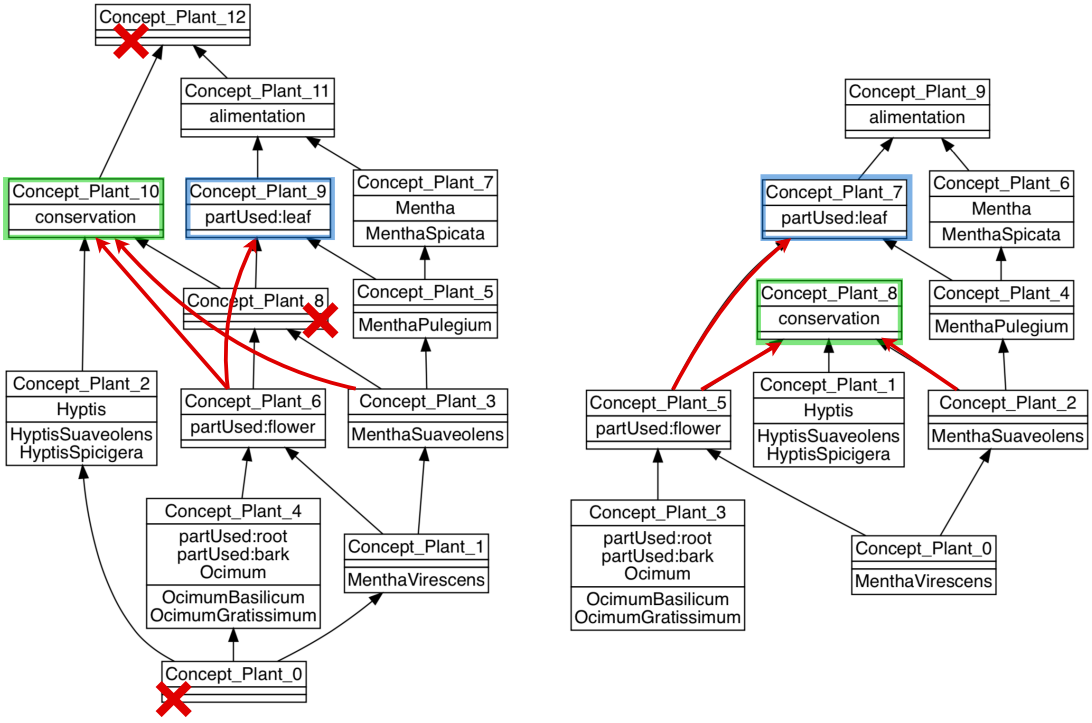


FIGURE 2.2 – *Treillis de concepts (gauche) versus AOC-poset (droite) construits pour le contexte formel $\mathcal{K}_{Plant}$ (Table 2.1b). Les concepts de $\mathcal{L}_{Plant}$ absents de l'AOC-poset sont repérés par une croix rouge.*

Définition 2.2.1. *L'AOC-poset (pour Attribute-Object-Concept poset) est le sous-ordre du treillis de concepts $\mathcal{L}_K$ restreint aux concepts introducteurs d'objet et d'attributs.*

Treillis Iceberg Un treillis Iceberg [Stumme et al., 2002] est un treillis de concepts duquel les concepts ayant un support inférieur à un seuil prédéfini $s \in [0, 1]$ ont été

retirés. Pour apparaître dans le treillis Iceberg, un concept doit donc être un concept « fréquent » et posséder un nombre d'objets supérieur ou égal à $s\%$ du nombre total d'objets. La figure 2.3 présente le treillis de concepts (à gauche) et le treillis Iceberg correspondant (à droite) avec le seuil $s = 50\%$. De même que pour la figure 2.2, les numéros des concepts de même extension diffèrent entre les deux structures qui ont été produites séparément. Les deux structures ont été produites en utilisant le contexte formel $\mathcal{K}_{Plant}$ (table 2.1b) qui possède un total de 8 objets. Dans le treillis Iceberg ne sont donc conservés que les concepts ayant au moins 4 objets dans leur extension. Tous les concepts *Concept_Plant_i* avec $i \in \{4, 1, 2, 6, 3, 5\}$ (symbolisés par une croix sur la figure) sont donc enlevés car ils concernent moins de 4 objets. Pour conserver la structure de treillis, un dernier concept est ajouté au bas de la structure. Ce concept permet également de conserver tous les attributs introduits dans les concepts qui ont été retirés (symbolisés par une pastille orange), i.e. *partUsed:flower* de *Concept_Plant_4* du treillis de concepts, *Hyptis* de *Concept_Plant_2*, etc.

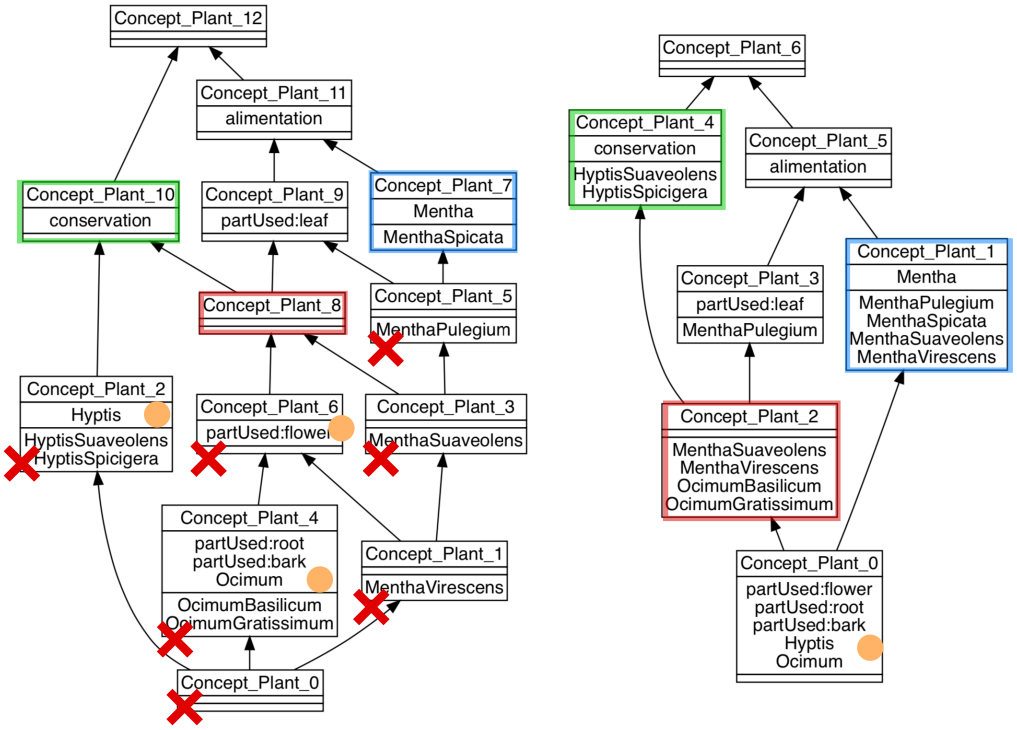


FIGURE 2.3 – *Treillis de concepts (gauche) versus Treillis Iceberg (droite). Seuls les concepts ayant un nombre d'objets supérieur à un seuil défini (ici 50%) sont présents dans le treillis Iceberg. Pour ce dernier, si nécessaire, un concept est ajouté au bas du treillis pour conserver la structure de treillis. Les attributs non introduits par les concepts présents y sont rassemblés. Dans le treillis $\mathcal{L}_{Plant}$, les concepts absents du treillis Iceberg sont indiqués avec une croix rouge ; les pastilles jaunes indiquent les attributs insérés dans le concept ajouté au bas de l'Iceberg.*

Définition 2.2.2. *Soit $B \subseteq M$, et soit $minsupp \in [0, 1]$. La valeur du support de B est $supp(B) = |\{g \in G \mid \forall m \in B, (g, m) \in I\}|/|G|$. B est appelé un ensemble d'attributs fréquent si $supp(B) >= minsupp$.*
Un concept est dit fréquent si son intension est fréquente.
Un treillis Iceberg est le sous-treillis du treillis de concepts $\mathcal{L}_K$ restreint aux concepts fréquents.

2.3 Analyse de Concepts Relationnels (RCA)

L'Analyse de Concepts Relationnels [Hacene et al., 2013] est une extension relationnelle de FCA. Elle permet de mettre en relation des groupes d'objets (des concepts) de contextes formels différents. Pour cela, elle prend en entrée une famille de contextes relationnels (RCF), notée $(\mathbf{K}, \mathbf{R})$, constituée de deux types de contextes différents. $\mathbf{K}$ représente l'ensemble des contextes formels ($\mathbf{K} = \{\mathcal{K}_i\}_{i=1,\ldots,n}$) chacun décrivant un type d'objets différent. $\mathbf{R}$ représente l'ensemble des contextes relationnels, i.e. l'ensemble des relations entre les objets des contextes de $\mathbf{K}$. $\mathbf{R} = \{r_j\}_{j=1,\ldots,p}$ et $r_j \subseteq G_k \times G_l$ pour $k, l \in \{1, \ldots, n\}$. Par exemple, reprenons pour $\mathbf{K}$ les contextes formels $\mathcal{K}_{Plant}$ et $\mathcal{K}_{Pest}$ de la Table 2.1, on peut définir pour $\mathbf{R}$ un contexte relationnel $r_{treatedBy}$ indiquant quels objets de $\mathcal{K}_{Pest}$ sont traités par les objets de $\mathcal{K}_{Plant}$. La table 2.2 illustre ce contexte relationnel $r_{treatedBy}$. La première colonne indique la source de la relation (i.e. les objets de $\mathcal{K}_{Pest}$), alors que la première ligne indique sa cible (i.e. les objets de $\mathcal{K}_{Plant}$). Dans cet exemple, on peut par exemple noter que *SpodopteraLittoralis* est traité par *MenthaSuaveolens* (illustré par les cellules rouges de la Table 2.2).

$r_{treatedBy}$	HyptisSuaveolens	HyptisSpicigera	MenthaPulegium	MenthaSpicata	MenthaSuaveolens	MenthaVirescens	OcimumBasilicum	OcimumGratissimum
CallosobruchusChinensis	×							
CallosobruchusMaculatus	×	×				×	×	×
RhipicephalusLunulatus							×	
RhipicephalusMicroplus								×
SpodopteraLittoralis			×	×	×	×	×	×

TABLE 2.2 – *Contexte relationnel $r_{treatedBy}$ décrivant les relations entre les objets de $\mathcal{K}_{Pest}$ et $\mathcal{K}_{Plant}$ (Table 2.1). Prises ensemble, les 3 tables constituent une RCF.*

La figure 2.4 illustre le mode opératoire de RCA pour calculer les structures conceptuelles. La couleur noire représente ce qui concerne les contextes formels et les éléments qui leur sont associés, i.e. les structures conceptuelles (treillis, AOC-posets, Iceberg) et les algorithmes de construction des structures. La couleur rouge représente les éléments relationnels, i.e. les contextes relationnels et des opérateurs de quantifications, dénommés *quantifieurs*, décrits ci-après.

À partir de la RCF, RCA construit les treillis pour chaque contexte formel de $\mathbf{K}$. Des attributs relationnels sont construits en utilisant chacun des contextes relationnels de $\mathbf{R}$. Ces nouveaux attributs sont de la forme $qr(\mathcal{C})$. q est un quantifieur inspiré des logiques de description (e.g. le quantifieur existentiel $\exists$). Le quantifieur est au préalable choisi par l'utilisateur en fonction du type d'information qu'il souhaite extraire. r est la relation représentée et $\mathcal{C}$ est le concept visé dans le treillis cible de la relation. Ensuite, les contextes formels sont étendus avec ces attributs relationnels. La table 2.3 montre le contexte formel $\mathcal{K}_{Pest}$ étendu par les attributs relationnels produits à partir de la relation $r_{treatedBy}$ en utilisant le quantifieur $\exists$.

La table 2.2, indique que *SpodopteraLittoralis* est traité par *MenthaSuaveolens*. RCA

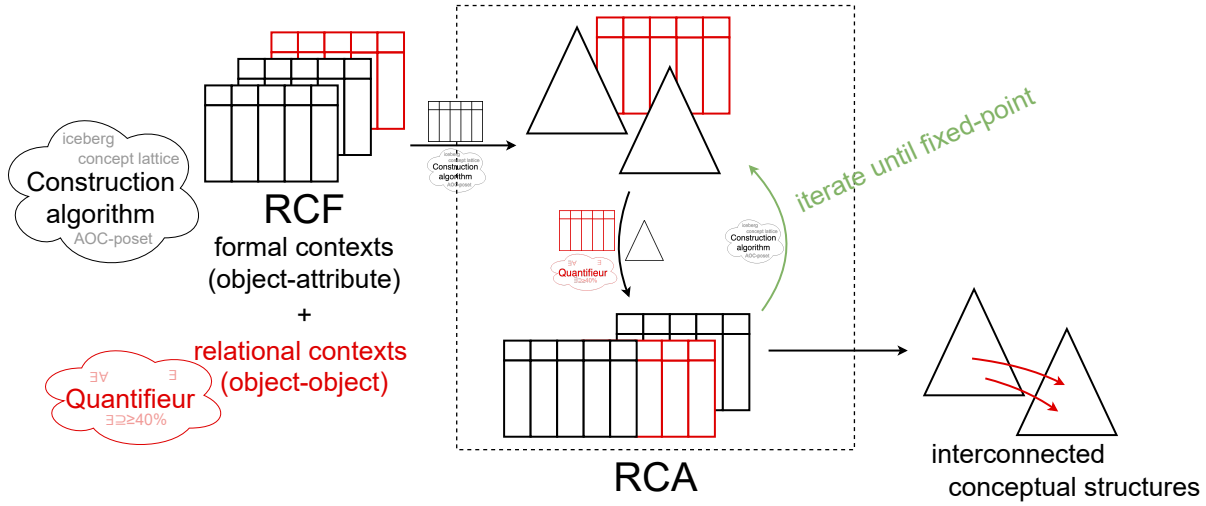


FIGURE 2.4 – *Mode opératoire de RCA pour calculer les structures conceptuelles. Dans un premier temps, RCA construit les structures conceptuelles (e.g. les treillis) correspondant à chaque type d'objets. Ensuite, les contextes formels sont étendus avec de nouveaux attributs nommés attributs relationnels. Ces attributs sont construits en utilisant les relations, les treillis et des quantifieurs inspirés des logiques de description (e.g. ∃). Enfin, de nouveaux treillis sont construits en utilisant les contextes formels étendus et ils sont connectés grâce aux attributs relationnels. On répète les deux dernières étapes jusqu'à atteindre un point fixe où les treillis ne sont plus modifiés.*

Tab 2.1a $\mathcal{K}_{Pest}$

$\mathcal{K}_{Pest}$	Callosobruchus	Rhipicephalus	Spodoptera	Burkina	Cameroun	Benin	goat	niebe	Animalia	∃ treatedBy(Concept_Plant_7)	∃ treatedBy(Concept_Plant_5)	∃ treatedBy(Concept_Plant_3)	∃ treatedBy(Concept_Plant_1)	∃ treatedBy(Concept_Plant_2)	∃ treatedBy(Concept_Plant_0)	∃ treatedBy(Concept_Plant_11)	∃ treatedBy(Concept_Plant_10)	∃ treatedBy(Concept_Plant_8)	∃ treatedBy(Concept_Plant_4)	∃ treatedBy(Concept_Plant_9)	∃ treatedBy(Concept_Plant_6)	∃ treatedBy(Concept_Plant_12)
CallosobruchusChinensis	×			×				×	×					×			×					×
CallosobruchusMaculatus	×				×			×	×	×	×	×	×	×		×	×	×	×	×	×	×
RhipicephalusLunulatus		×			×	×	×		×							×	×	×	×	×	×	×
RhipicephalusMicroplus		×			×	×	×		×							×	×	×	×	×	×	×
SpodopteraLittoralis			×		×	×			×	×	×	×	×			×	×	×	×	×	×	×

TABLE 2.3 – *Le contexte formel $\mathcal{K}_{Pest}$ étendu par les attributs relationnels produits à partir de la relation $r_{treatedBy}$. Le contexte formel original $\mathcal{K}_{Pest}$ est ici représenté entouré de pointillés. Toutes les nouvelles colonnes représentent les attributs relationnels créés à partir de $r_{treatedBy}$. Les cellules rouges représentent les attributs relationnels de l'objet SpodopteraLittoralis.*

crée donc des attributs relationnels dans le contexte $\mathcal{K}_{Pest}$ menant vers les concepts contenant *MenthaSuaveolens*, i.e. vers les *Concept_Plant_i*, avec i ∈ {3,5,7,8,9,10,11,12}. En effet, *MenthaSuaveolens* est introduit dans le concept 3 de $\mathcal{L}_{Plant}$ et il est hérité vers le haut dans tous les super-concepts de *Concept_Plant_3*. Les nouveaux attributs sont ∃ *treatedBy(Concept_Plant_i)* avec i ∈ {3,5,7,8,9,10,11,12}. Pour faciliter la lecture, les attributs relationnels sont notés *q relation(Concept)* au lieu de *q* $r_{relation}$*(Concept)* dans la suite du document. Ces nouveaux attributs font partie des attributs relationnels ajoutés dans la table 2.3 et sont connectés à *SpodopteraLittoralis*. L'utilisation du quantifieur exis-

tentiel implique que *SpodopteraLittoralis* est traité par au moins un des objets de chaque concept ciblé.

Une fois tous les contextes relationnels étendus par l'ajout des attributs relationnels, des treillis sont à nouveau produits sur ces contextes. Ce processus (extension des contextes et génération des nouveaux treillis) est répété jusqu'à ce qu'un point fixe soit atteint, i.e. jusqu'à ce que les contextes (et donc les treillis) ne changent pas entre deux itérations. La figure 2.5 présente les treillis produits par RCA sur la RCF constituée des contextes formels $\mathcal{K}_{Pest}$ et $\mathcal{K}_{Plant}$ et du contexte relationnel $r_{treatedBy}$. Le fait que *SpodopteraLittoralis* soit traité par *MenthaSuaveolens* est représenté en particulier par l'attribut relationnel $\exists$ *treatedBy(Concept_Plant_3)* qui indique qu'il est traité par au moins une plante du concept 3. Cela ne signifie pas que *SpodopteraLittoralis* est traité par *MenthaVirescens*. Mais cela supporte l'hypothèse que cela pourrait être le cas. L'utilisation du quantifieur existentiel introduit donc des relations additionnelles entre les objets.

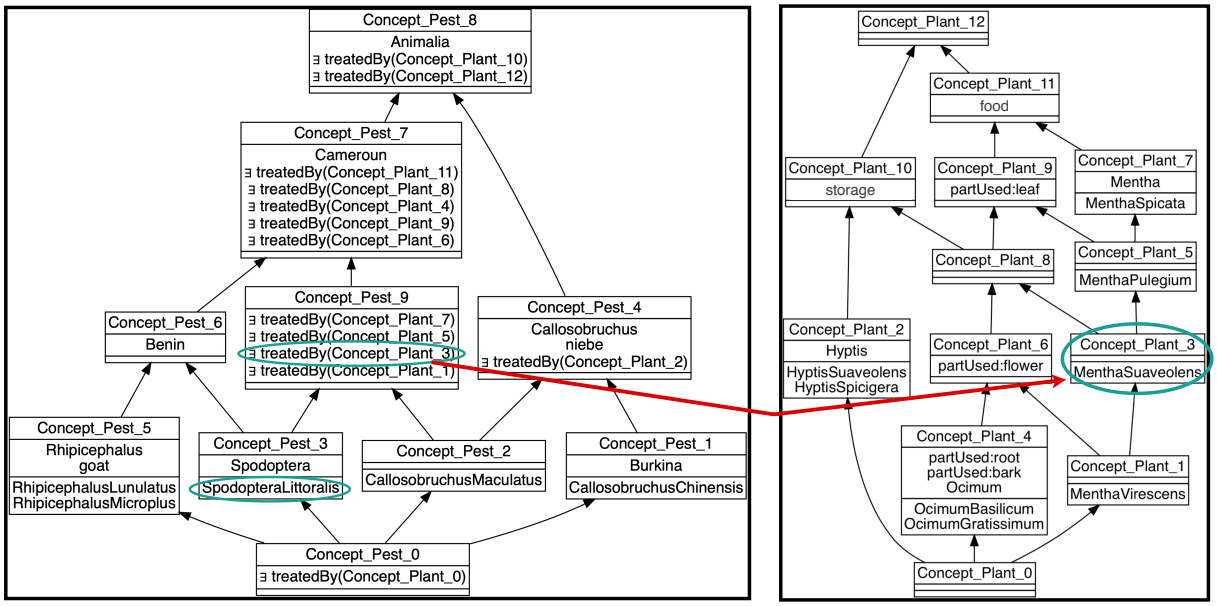


FIGURE 2.5 – *Famille de treillis produite avec RCA sur la RCF. A gauche : $\mathcal{L}_{Pest}$. A droite : $\mathcal{L}_{Plant}$. Dans le Concept_Pest_3 de $\mathcal{L}_{Pest}$, l'objet SpodopteraLittoralis possède l'attribut $\exists$ treatedBy*(`Concept_Plant_3`) *hérité du Concept_Pest_9. Cela signifie que SpodopteraLittoralis est traité par au moins un objet du Concept_Plant_3, i.e. MenthaSuaveolens ou MenthaVirescens.*

Par ailleurs, si on compare le treillis $\mathcal{L}_{Pest}$ avec sa version avant l'ajout des attributs relationnels, on constate la présence de nouveaux concepts. Par exemple, *Concept_Pest_9* n'existait pas dans le treillis original (voir figure 2.1). Il a été ajouté car il regroupe *SpodopteraLittoralis* et *CallosobruchusMaculatus*, qui partagent désormais le fait d'être traités par au moins une plante de *Concept_Plant_3*. En effet, les plantes *MenthaSuavolens* et *MenthaVirescens*, qui traitent respectivement *SpodopteraLittoralis* et *CallosobruchusMaculatus*, sont toutes deux présentes dans le *Concept_Plant_3*.

2.4 Triadic concept analysis (TCA)

L'analyse triadique de concepts (TCA) est une extension de FCA permettant de traiter les situations du genre « *an object g has the attribute m under the condition b* » [Lehmann and Wille, 1995]. Là où un contexte formel de FCA est un 3-tuple $\mathcal{K} = (G, M, I)$, le contexte triadique est un 4-tuple $K = (G, M, B, Y)$, et un concept triadique est un triplet d'ensembles $(A1, A2, A3)$ (voir Définition 2.4.1).

Définition 2.4.1. *Un contexte triadique est un 4-tuple $K = (G, M, B, Y)$ où G est l'ensemble des objets, M l'ensemble des attributs, B l'ensemble des conditions et où $Y \subseteq G \times M \times B$ associe un objet et un attribut sous une condition.*
Un concept triadique est un triplet $(A1, A2, A3)$ où $A1 \subseteq G$, $A2 \subseteq M$, $A3 \subseteq B$ et $A1 \times A2 \times A3 \subseteq Y$.

Comparativement à FCA, on peut dire que TCA utilise des contextes formels, chacun ayant des objets, des attributs et une condition, afin de former un contexte tridimensionnel, comme illustré par la figure 2.6. Un concept issu de FCA peut être représenté comme une matrice rectangulaire dont chaque case contient une croix (×). Dans une représentation tridimensionnelle, un concept triadique peut, par généralisation, être représenté par une matrice tridimensionnelle dont chaque cube contient une croix (×). Ceci est symbolisé par les cases en rouge dans la figure 2.6.

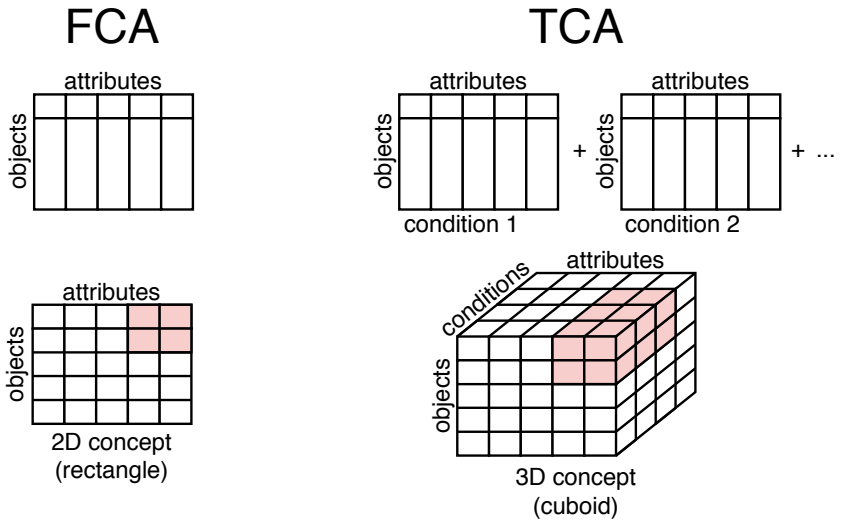


FIGURE 2.6 – *Représentation d'un contexte et d'un concept (en rouge) pour FCA et pour TCA.*

Si on considère les ensembles d'attributs et de conditions comme deux autres ensembles d'objets quelconques, on peut ainsi étendre l'utilisation de TCA aux relations ternaires entre 3 catégories d'objets. De cette façon, une croix dans le contexte triadique représenterait une relation ternaire entre 3 objets A1, A2 et A3.

2.5 Analyse conceptuelle de graphes de connaissance (Graph-FCA ou GFCA)

Graph-FCA (ou GFCA) [Ferré and Cellier, 2016] est une extension de FCA permettant de traiter les graphes de connaissances, i.e. des données représentées sous forme de graphe. Un *graph context* est un 3-tuple $K = (O, A, I)$ permettant de représenter des relations d'incidence entre des tuples d'objets et d'attributs d'après la définition 2.5.1.

Définition 2.5.1. *Un graph context est un triplet $K = (O, A, I)$ où O est l'ensemble des objets, A est l'ensemble des attributs et $I \subseteq O^* \times A$ est une relation d'incidence entre des tuples d'objets ($o \in O^*$) et d'attributs ($a \in A$).*

La figure 2.7 illustre un *graph context* de GFCA (à droite) comparativement à un contexte formel de FCA (à gauche). Les objets du *graph context* sont représentés par les carrés (i.e. o1, o2 et o3). Les attributs sont des relations, qu'il s'agisse de relations entre les objets (b1 et c1) ou des attributs plus classiques (tels qu'avec FCA) qui sont ici considérés comme des relations unaires (i.e. a1, a2 et a3). Les éléments de I sont les arêtes du graphe, i.e. ((o1,o2),b1), ((o2,o3),b1) et ((o1,o2, o3),c1). Par exemple, ((o1,o2),b1) est l'arête qui relie les objets o1 et o2 par la relation b1.

	a1	a2	a3
o1	x		x
o2	x	x	
o3		x	

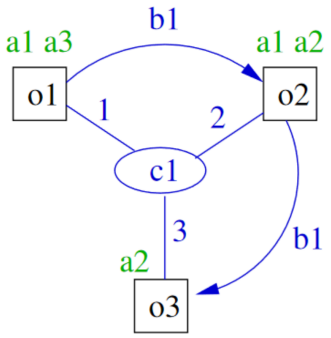


FIGURE 2.7 – *GFCA : Représentation d'un graph context de GFCA (à droite) par rapport à un contexte de FCA (à gauche). Les relations entre les objets (b1 et c1) font également partie de l'ensemble des attributs.*

En utilisant ainsi les relations en tant qu'attributs, GFCA produit, comme FCA, des concepts qui sont organisés par un treillis de concepts. L'extension de ces concepts est un ensemble d'objets. Leur intension est ce qu'on appelle un *graph pattern*. Les *graph patterns* sont des généralisations des incidences du *graph context*. La figure 2.8 illustre un *graph pattern* où x et y sont deux objets quelconques de O. Les attributs du *graph pattern* sont a1, a2 et b1. Les éléments d'incidence sont : ((x), a1), ((y), a2) et ((x, y),b1). Cela signifie respectivement que : x possède l'attribut a1, y possède l'attribut a2, et x est en relation avec y par la relation b1.

Pour des questions de lisibilité, dans la représentation graphique du treillis, l'intension des concepts n'est pas écrite dans chaque concept du treillis. Elle est écrite à côté des éléments dans le treillis. La figure 2.9 illustre le treillis de concepts produit par GFCA à partir du *graph context* de la figure 2.7. On y retrouve le *graph pattern* de la figure 2.8 dans le haut du treillis. En plus de superposer les graph patterns sur le treillis, ces

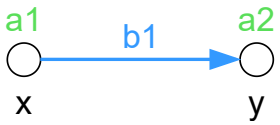


FIGURE 2.8 – *GFCA : Représentation d'un graph pattern où* $x, y \in O$, $a1, a2, b1 \in A$ *et* $((x), a1), ((y), a2), ((x, y)b1) \in I$.

derniers sont également hérités vers le bas. Ainsi, on sait que les concepts introduisant o2 et o3 sont en relation par b1 (de o2 vers o3).

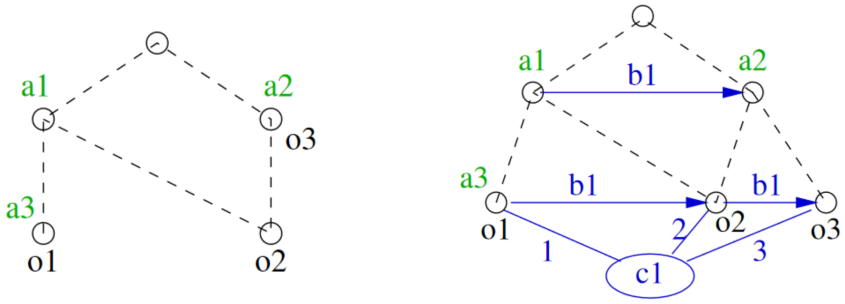


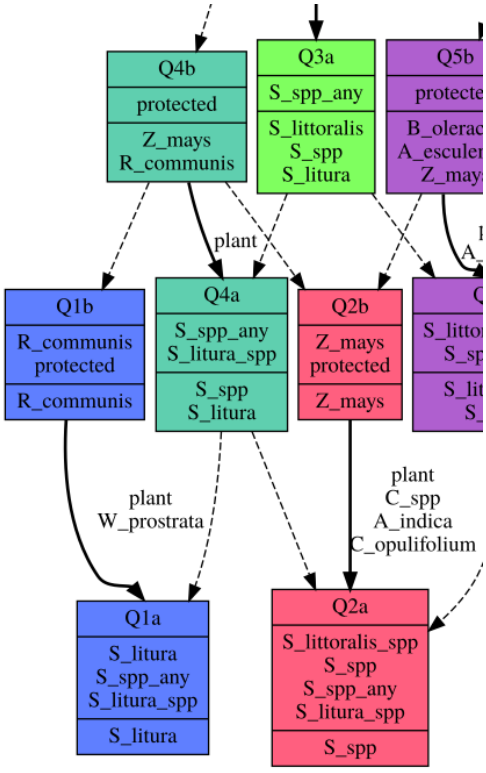
FIGURE 2.9 – *GFCA : Treillis de concepts produit à partir de la figure 2.7. Les graph patterns (comme celui de la figure 2.8) sont projetés directement sur le treillis.*

La figure 2.10 illustre la spécification de l'intension de deux concepts sur un extrait de treillis de concepts produit par GFCA. Pour ce faire, seuls les attributs sont représentés sur la partie droite de la figure. Les deux concepts verts foncé (Q4b et Q4a) indiquent que Z.mays et R.communis (des organismes protégés) sont protégés par une plante (sans indiquer laquelle) contre S.litura et S.spp. Ces concepts, mais également la relation qui les relie, sont spécialisés de deux façons, d'un côté avec les concepts bleus (Q1b et Q1a) et de l'autre avec les concepts rouges (Q2b et Q2a). Les concepts rouges spécifient le pattern vert en indiquant que Z.mays est protégé contre S.spp. en utilisant l'une des 3 plantes C.spp, A.indica et C.opulifolium.

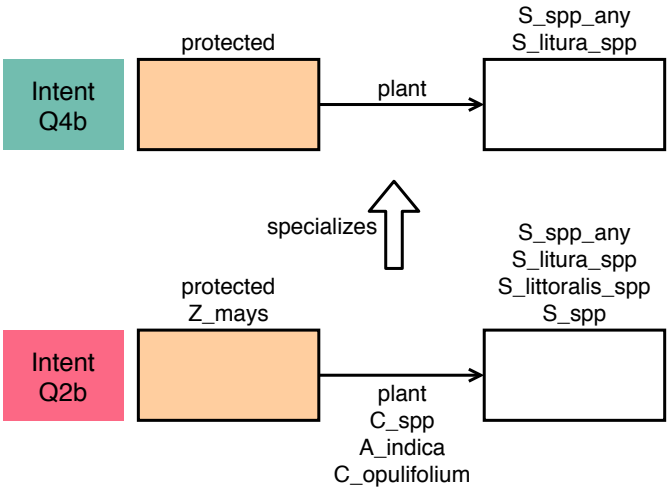
2.6 Scaling

Le scaling est une opération qui binarise une donnée multi-valuée. Le scaling d'un attribut multi-valué [Ganter and Wille, 1999, Sebastian, 2021] est obtenu en utilisant un contexte formel dérivé. Pour chaque contexte dérivé, la partie gauche (celle des objets) est remplie avec les différentes valeurs possibles de l'attribut. La partie du haut (celle des attributs) est remplie avec des attributs booléens décrivant les valeurs d'une manière qui sera utile pour l'analyse. Le scaling d'un attribut consiste à indiquer, pour chaque valeur d'attribut, le ou les attributs booléens correspondants.

Nous illustrons différents types de scaling. La table 2.4a illustre le scaling nominal, où chaque valeur de l'attribut multi-valué est décrite par un seul attribut booléen. Dans cet exemple, les objets A, B et C sont décrits respectivement par SpeciesA, SpeciesB ou SpeciesC. La table 2.4b illustre le scaling ordinal, où, dans la mesure où les valeurs sont classées hiérarchiquement, chaque valeur de l'attribut multi-valué est mis en correspon-



(a) *Extrait d'un treillis produit par GFCA.*



(b) *L'intension des concepts rouges spécifie l'intension des concepts verts.*

FIGURE 2.10 – *Illustration de la spécialisation de l'intension de concepts dans un treillis produit par GFCA. L'intension contient les attributs au sens de FCA mais également les relations telles que la relation plant.*

dance avec sa description. Dans l'exemple proposé, 2010 étant plus grand que 2000, la valeur 2010 est décrite par ≥2000 et ≥2010. La table 2.4c illustre une variante du scaling ordinal que nous appelons *scaling par intervalles*. Cette dernière version du scaling utilise des intervalles de valeurs bornés qui peuvent se chevaucher. Il se distingue du scaling bi-ordinal de [Ganter and Wille, 1999] qui utilise des combinaisons d'attributs ≥ et ≤), ainsi que du scaling interordinal de [Kaytoue et al., 2011] qui correspond à un scaling bi-ordinal.

species	speciesA	speciesB	speciesC
A	×		
B		×	
C			×

(a) *Scaling nominal*

year	≥2000	≥2010	≥2020
2000	×		
2010	×	×	
2020	×	×	×

(b) *Scaling ordinal*

year	[2000,2010]	[2005,2010]	[2000,2005]	[2000,2020]
2002	×		×	×
2005	×	×	×	×
2019				×

(c) *Scaling par intervalles*

TABLE 2.4 – *Exemples de scaling d'attributs pour obtenir des attributs booléens. (a) Nominal : chaque valeur d'un attribut est convertie en un attribut booléen spécifique. (b) Ordinal et (c) par intervalles : chaque valeur est décrite par un ou plusieurs attributs.*

2.7 Conclusion

Ce chapitre a introduit FCA (section 2.1), différentes structures conceptuelles que peut produire FCA (section 2.2), ainsi que des extensions relationnelles de FCA, i.e. RCA (section 2.3), TCA (section 2.4) et GFCA (section 2.5). Ce chapitre s'est terminé en présentant le scaling, i.e. la méthode utilisée avec FCA pour transformer les attributs non booléens (section 2.6).

Le chapitre suivant présente des travaux représentatifs sur la conversion des connaissances dans différents domaines de l'Informatique : les bases de données, le génie logiciel, la science des données et la représentation des connaissances.

III

État de l'art sur la conversion des connaissances

La conversion des connaissances occupe une part importante des travaux de recherche dans plusieurs domaines de l'informatique. Comme ces domaines sont larges, le chapitre détaille seulement quelques articles représentatifs de ces questions pour quatre domaines qui sont particulièrement concernés par cette question : les bases de données (section 3.1), le génie logiciel (section 3.2), la science des données (section 3.3) et la représentation des connaissances (section 3.4) au moyen des ontologies (section 3.4.1) et de l'Analyse de Concepts Formels (section 3.4.2). Enfin la section 3.5 présente une synthèse des documents abordés. La section 3.6 conclut ce chapitre.

3.1 Conversion dans le domaine des bases de données

Les bases de données permettent de stocker des données structurées ou non structurées. Les données d'une base peuvent provenir d'une ou plusieurs autres bases ou d'autres sources de données (collectées en ligne, textes ou provenant de fichiers variés). Dans le premier cas, il faut convertir les données de la ou des bases sources en s'appuyant sur leur schéma pour correspondre à la base cible. Dans le second cas, il faut tenir compte des caractéristiques des connaissances réelles pour le stockage. Certaines situations seront complexes, comme par exemple le cas où les données sont organisées dans des relations n-aires. Pour illustrer les problèmes de conversion et d'intégration dans les bases de données, cette section présente deux types de travaux : les principes des ETL (Extract Transform Load) et les méthodes de conversion d'une relation n-aire.

Extract Transform Load Les logiciels ETL permettent l'intégration de données, consistant à collecter et à transférer des connaissances provenant d'une ou plusieurs sources de données vers une base de données ou un *Data Warehouse*. Leur processus consiste d'abord

à extraire les données des sources d'origine, puis à les transformer pour correspondre au format de la cible et enfin à les y charger. Selon [Trujillo and Luján-Mora, 2003], le processus ETL se décompose en 6 étapes. La première consiste à sélectionner les données sources à transférer et qui sont en provenance d'une ou plusieurs sources (qui sont souvent des bases de données). La deuxième est la transformation de chaque source de données via l'utilisation de filtres, le calcul de valeurs dérivées, le changement de format, etc. La troisième étape fusionne les différentes sources, qui sont ensuite chargées ensemble dans la base cible sélectionnée lors de la quatrième étape. La cinquième étape fait le lien entre sources et cible en projetant les données sources transformées sur les champs de la base cible. Enfin, la sixième étape charge les données dans la base cible. L'article présente plusieurs transformations des attributs qui ont lieu lors de la deuxième étape du processus décrit, notamment l'agrégation et la conversion. L'agrégation de valeurs calcule des valeurs dérivées en fusionnant des données avec des fonctions telles que la somme, la moyenne, le maximum ou le minimum, etc. La conversion regroupe plusieurs opérations : conversion de type (transformation d'une chaîne de caractères en nombre entier), conversion arithmétique (avec des opérations arithmétiques, e.g. addition, multiplication), conversion de chaîne de caractères (concaténation, passage en majuscule, etc.), conversion de division (e.g. division d'une chaîne de caractères), conversion de standardisation (uniformisation de valeurs telles que « jan. » ou « 1 » en « January »), initialisation d'une valeur avec une valeur par défaut, etc. L'explication des opérations par l'article et l'illustration sur des exemples permettent notamment de comprendre l'effet qu'auront les transformations des attributs sur les données.

L'article [Skoutas et al., 2009] présente une approche ETL basée sur des graphes orientés pour convertir les données d'un graphe source vers un graphe cible. Dans cette approche, les nœuds du graphe représentent les éléments du schéma de base (catégorie d'objet ou attribut) et les arêtes représentent les relations entre ces éléments (relation objet-attribut ou objet-objet). En fonction du niveau de similarité entre les graphes source et cible, différentes actions sont effectuées sur le graphe. L'article illustre notamment plusieurs actions effectuées sur les attributs. Par exemple, ces derniers peuvent être transférés directement dans le nouveau graphe s'ils correspondent à la cible, ou un attribut peut être scindé en plusieurs attributs (e.g. le nom « John Doe » peut être décomposé en deux attributs nom « Doe » et prénom « John ») ou inversement fusionner plusieurs attributs. L'article montre ainsi des modifications au niveau des attributs mais pas sur l'organisation des classes entre elles.

La conversion de relations N-aires Les données décrivent des situations. Pour les décrire, des verbes sont utilisés qui relient les différentes données. Chaque verbe représente donc une relation entre les données, et la valence du verbe désigne l'arité de la relation. Par exemple, dans les données que nous traitons, le verbe « protège » est de valence 3 car une plante (1) protège un organisme (2) contre un agresseur (3). Ainsi, en utilisant une arité correspondant à la valence du verbe, on évite de faire des modifications dans la relation. L'utilisation d'une relation n-aire permet donc de mieux refléter la signification d'une situation. Néanmoins, certains systèmes d'information ainsi que certains systèmes de gestion de données utilisent majoritairement des modèles de représentation ne donnant que la possibilité de représenter des relations binaires. Il est donc nécessaire de transformer les relations n-aires en relations binaires. Ce problème a été abordé par les recommanda-

tions du W3C [Noy and Rector, 2006]. Dans ce document, les auteurs mettent l'accent sur la réification d'une relation n-aire, avec « une nouvelle classe et n nouvelles propriétés ». L'article [Jones and Song, 2000] présente également plusieurs alternatives à la réification en fonction de la cardinalité associée à chaque extrémité d'une relation ternaire. En étudiant les relations logiques implicites à la relation ternaire, les auteurs identifient ainsi les décompositions binaires équivalentes possibles.

3.2 Le refactoring en Génie Logiciel

Le génie logiciel (ou ingénierie logicielle) est un domaine qui étudie les méthodes menant à la conception et à la réalisation de logiciels. Il s'intéresse notamment aux procédures utilisées pour développer, maintenir, tester et évaluer les logiciels. Comme illustration des conversions effectuées dans ce domaine, nous avons choisi le refactoring. La technique du refactoring occupe une place importante dans ce domaine. Elle consiste à retravailler le modèle ou le code source d'un logiciel pour améliorer sa lisibilité ou en réutiliser des parties. Une meilleure lisibilité permet une meilleure compréhension et une maintenance plus efficace du modèle ou du code source. La réutilisation fait économiser du temps et évite des répétitions. Une autre application du refactoring consiste à migrer le code source écrit dans un langage devenu obsolète en code source développé dans un langage plus récent, pour lequel il sera plus simple de trouver des personnes qualifiées. Cette section présente différents travaux sur le refactoring.

Le livre [Fowler, 2018] regroupe un catalogue d'opérations de transformation sur des classes, associations, attributs ou méthodes. Ces opérations permettent la migration et l'optimisation de code, par exemple pour une meilleure maintenance. Elles n'ont donc pas pour but initial de modifier le modèle. Néanmoins, afin d'obtenir une meilleure architecture du code, le modèle peut être modifié.
N'ayant pas de code (donc pas de méthodes) dans nos modèles, les opérations qui les concernent ne nous intéressent pas. Parmi les autres opérations, celles qui sont pertinentes par rapport à nos besoins sont les suivantes :

- « pull up field »: supprime un attribut d'une sous-classe pour le mettre dans la super-classe
- « push down field »: supprime un attribut d'une super-classe pour le mettre dans certaines sous-classes
- « change bidirectional to unidirectional »: transforme une association bidirectionnelle de façon à ne conserver qu'une seule des deux directions de l'association (dans le cas où l'une des deux classes n'exploite pas sa relation avec l'autre)
- « collapse hierarchy »: fusionne une sous-classe avec sa super-classe en remontant les attributs de la sous-classe avec « pull up field » et ses méthodes (avec « pull up method ») dans la super-classe
- « replace subclass with fields »: introduire, dans la super-classe, un attribut indiquant l'appartenance à la sous-classe. Cet attribut remplace une méthode présente dans une sous-classe retournant une valeur identifiant la sous-classe. Par exemple, on peut remplacer la sous-classe Femme de la classe Personne et sa méthode retournant « F » par un attribut `genre` dans la classe Personne.

Dans le cadre de ce mémoire, certaines de ces opérations sont utiles en l'état, à l'exemple de *pull up field et push down field*. D'autres, en revanche, sont intéressantes mais doivent être enrichies ou modifiées. Dans le cas général, la transformation d'une association bidirectionnelle en une unique association unidirectionnelle ne préserve pas la sémantique. Pour préserver la sémantique, il faudrait transformer l'association bidirectionnelle en deux associations unidirectionnelles. Pour fusionner une sous-classe avec sa super-classe dans du code source, il suffit de remonter les attributs et les méthodes car les associations sont présentes en tant qu'attributs. Au niveau modèle, les associations peuvent être présentes sous forme d'attributs ou de liens entre les deux classes. Il faut donc également faire remonter ces liens au niveau de la super-classe. Pour notre problématique de conversion, nous avons besoin de faire disparaître des sous-classes et nous pouvons utiliser le principe de l'opération « replace subclass with fields ». Par ailleurs, les opérations décrites dans [Fowler, 2018] ne tiennent pas compte des instances. De plus, même si toutes les opérations définies sont décrites, ce ne sont que des modifications locales. Il existe également des refactorings plus imposants, ce que [Fowler, 2018] nomme des *big refactorings*, e.g. la séparation des aspects de visualisation et de programmation qui étaient fusionnés dans les mêmes classes dans un logiciel. Les *big refactorings* demandent d'appliquer plusieurs modifications qui peuvent dépendre les unes des autres. Par exemple, on ne peut remonter un attribut dans une super-classe que si la super-classe a été créée. Ces *big refactorings* demanderont d'avoir un algorithme ordonnant les applications des différentes opérations.

L'article [Sunyé et al., 2001] présente des exemples de refactoring sur un modèle UML qui préservent le comportement du modèle. Cet article s'inscrit dans le domaine de l'Ingénierie Dirigée par les Modèles où les transformations de modèles sont nombreuses. Celles de [Sunyé et al., 2001] sont des transformations basées sur le principe *« Add, Remove, and Move »*. Les auteurs expliquent comment, dans certaines conditions, l'ajout, la suppression ou le déplacement d'éléments dans un diagramme de classes UML ne modifient pas le comportement du système. Ainsi, en combinant ces actions *« Add, Remove, and Move »*, on peut retrouver les transformations de « pull up field » et « collapse hierarchy » présentes dans [Fowler, 2018].

3.3 La construction de jeux de données en science des données

La science des données est l'étude de l'extraction généralisable de connaissances à partir de données [Dhar, 2013]. Pour ce faire, la construction de jeu de données à destination des différentes méthodes possibles est indispensable, comme le montre l'article [de la Vega et al., 2020]. Cet article présente un langage visant à automatiser certaines des tâches de gestion des données que les *data scientists* doivent effectuer lors de la création d'ensembles de données. Il comprend un ensemble de fonctions permettant de transformer les données. Les fonctions disponibles, qui consistent à construire une table de données, sont les suivantes :

- Construction d'une table par sélection d'une classe et de tous ses attributs (par

défaut). Un attribut devient une colonne et une instance devient une ligne.
- Construction d'une table par sélection d'une classe et d'une partie de ses attributs ou par filtrage des instances sur des valeurs d'attributs.
- Inclusion, dans la table associée à une première classe sélectionnée, d'informations provenant d'une seconde classe, en suivant les références de la première classe, par ajout de colonnes pour tout ou partie des attributs de la seconde classe.
- Variante de la fonction d'inclusion précédente, dans laquelle les informations incluses proviennent d'une ou plusieurs classes connectées par héritage avec la seconde classe et de tout ou partie de leurs attributs.

La gestion de l'héritage est similaire au *Single table inheritance pattern* de [Fowler, 2012] et n'est pas sans rappeler les techniques du Génie Logiciel utilisées dans [Fowler, 2018]. Dans la mesure où les associations sont considérées via les références d'une première classe à une seconde classe, cette approche ne considère que des associations binaires navigables dans une seule direction.

3.4 La conversion appliquée au domaine de la représentation des connaissances

Nous commençons par définir le cadre que pose la représentation des connaissances. « *La représentation formelle et symbolique des connaissances repose sur des langages formels composés d'une part de règles syntaxiques génératives pour la construction de formules et d'autre part de règles de composition sémantique associant aux formules construites une signification. Le problème posé par la représentation des connaissances est de formaliser dans un langage formel de représentation les connaissances permettant de traiter le problème à résoudre* » [Bachimont, 2000]. Pour illustrer la conversion dans ce domaine, nous nous intéressons aux conversions liées aux ontologies ou à la préparation de données pour l'Analyse de Concepts Formels.

3.4.1 Conversion et extraction d'une ontologie

Une ontologie est un ensemble structuré de termes et de concepts permettant de représenter un domaine particulier. Elle utilise principalement des classes (ou concepts), des attributs, des individus et des relations. Le langage OWL (Web Ontology Language) est un langage couramment utilisé pour représenter une ontologie. OWL est un langage basé sur le modèle RDF (Resource Description Framework) [Wood et al., 2014]. Ce langage est très couramment utilisé par la communauté du web sémantique, tout comme UML (Unified Modeling Language) pour celles des bases de données et du génie logiciel. De part le lien ténu existant entre ces communautés, il est courant de voir des conversions entre leurs modes de représentations. Par ailleurs, il existe également des méthodes permettant de produire une ontologie à partir d'une base de données afin d'en obtenir un niveau d'abstraction supplémentaire ([Calvanese et al., 2021]). Cela permettra ensuite, par exemple, de faire évoluer plus facilement la base de données concernée.

Conversion d'un diagramme de classes UML vers une ontologie OWL L'article [Bouihi and Bahaj, 2019] propose une approche de conversion entre un diagramme de classes UML et une ontologie au format OWL. Cette approche utilise des opérations présentées dans [Bouihi and Bahaj, 2016] qui s'appliquent à des classes, des instances, des attributs et des associations. Parmi celles-ci, nous pouvons notamment retenir les suivantes qui sont aisément transposables pour notre propos :

- Classe → Classe
- Instance → Individu
- Attribut → Functional Datatype Property qui transforme un attribut UML en un attribut OWL ayant au plus une valeur
- Association → ObjectProperty avec une spécification du domaine (source) et du co-domaine (cible) qui transforme une association (binaire) en une relation entre deux classes source et cible

En plus de ces opérations, des cas particuliers sont également proposés. L'identifiant d'une classe est converti en « Inverse-Functional Property ». Il s'agit d'un attribut qui correspondrait à une clef primaire d'une table dans une base de données. Dans notre cas, nous n'avons pas d'identifiant sur nos classes. Cependant, si nous en avions, nous pourrions les traiter de la même façon que tous les autres attributs. Concernant les associations, pour notre contexte (FCA/RCA) qui considère des associations binaires simples, les transformations concernant les autres types d'associations tels que les agrégations, compositions ou dépendances ne sont pas applicables. Enfin, le concept de l'héritage est commun aux deux format UML et OWL, alors que ce n'est pas le cas pour notre contexte.
Avec le langage OWL, seules les associations binaires sont considérées et les associations n-aires doivent être converties comme avec FCA/RCA. Malheureusement ces conversions ne sont pas présentées dans l'article. Les attributs peuvent être multi-valués, il n'est donc pas nécessaire d'avoir des opérations de *scaling*.

Extraction d'ontologie L'article [Calvanese et al., 2021] présente un algorithme permettant d'extraire un modèle ontologique à partir d'un modèle de base de données. Pour ce faire, les auteurs utilisent différents patterns pour produire des éléments du langage OWL à partir des caractéristiques des tables présentes dans la base de données. Les patterns suivants sont abordés :

- « Schema Entity » (SE) : extrait une classe OWL à partir d'une table. La table est implémentée sous forme de classe, la clef primaire permet la création des instances de la classe et les attributs de la table forment les attributs de la classe.
- « Schema Relationship » (SR) : permet, à partir d'une configuration de trois tables, de construire une relation entre deux classes. Deux des trois tables deviennent des classes et la dernière devient une *Object property* (une relation) entre les deux classes précédentes.
- « Schema reified Relationship » (SRR) : crée une relation n-aire sous forme réifiée (matérialisée).

On notera également la présence du pattern « Schema hierarchy » (SH) qui produit une relation de hiérarchie entre deux tables et donc entre les deux classes OWL correspon-

dantes.

3.4.2 Préparation des données pour l'Analyse de Concepts Formels

Le problème de préparation des données pour FCA est bien connu dans le domaine. Il existe donc déjà des travaux théoriques mais aussi appliqués sur ce sujet et notamment sur la conversion des attributs multi-valués en attributs booléens. Comme nous l'avons vu, cette méthode de conversion est appelée le *scaling* des attributs.

***Scaling* d'attributs multi-valués** Le *scaling* d'attribut est une technique qui permet d'obtenir, à partir d'un attribut multi-valué, un ensemble d'attributs booléens. Ces attributs booléens sont utilisés dans les contextes formels pour décrire les caractéristiques des objets. Le livre [Ganter and Wille, 1999] pose les bases de l'Analyse de Concepts Formels et décrit différents types de *scaling* utilisés pour produire ces attributs booléens. Deux scaling principaux y sont décrits :

- « nominal »: crée un attribut booléen pour chacune des valeurs de l'attribut multi-valué
- « ordinal »: crée des attributs en fonction de valeurs numériques en utilisant un unique comparatif $\leq$ ou $\geq$ (e.g. « ≤ 2000 » et « ≤ 2010 » pour des dates)

Grâce à ces deux *scaling*, il est ensuite possible de définir d'autres *scaling* adaptés aux besoins de l'utilisateur. Le *scaling* ordinal est, par exemple, décliné en deux autres *scaling*. Le *scaling* interordinal permet de combiner l'utilisation des deux opérations de comparaison $\leq$ et $\geq$ (e.g. 2 est ≤ 4 et ≥ 1). Le *scaling* biordinal combine également les deux opérations de comparaison mais dans son cas, les intervalles ne se superposent pas (e.g ≤ 4 et ≥ 5). Enfin, le *scaling* dichotomique est un cas particulier combinant le *scaling* nominal et le *scaling* biordinal. Il est utilisé pour les attributs possédant les valeurs oui et non.

L'article [Neznanov et al., 2013] présente FCART, un système logiciel basé sur FCA pour la recherche d'informations. Les données sont présentées sous la forme d'une table, dans un fichier texte (e.g. un fichier au format CSV ou CXT). Les données peuvent être binaires, multi-valuées, des champs de texte, ou des ensembles de valeurs. À partir de cette table, le système binarise les attributs multi-valués en utilisant un ensemble de règles afin de générer un contexte formel. Parmi ces règles, on trouve :

- La « simple rule »: pour générer un attribut booléen à partir d'un unique attribut binaire de la table.
- La « scaling rule »: pour générer plusieurs attributs à partir d'un unique attribut multi-valué de la table grâce à des scaling nominaux ou ordinaux.

L'article ne précise pas la façon de produire les objets du contexte formel. On peut cependant supposer qu'une ligne de la table devient un objet du contexte formel. Comme cette méthode a pour objectif de transformer une table en un unique contexte formel, elle

ne considère pas la notion d'association.

Scaling pour des données taxinomiques L'article [Cellier et al., 2008] explique que la taxinomie décrit une relation d'ordre entre des attributs. L'article prend notamment l'exemple d'un objet possédant, dans sa description, l'attribut « Floride » mais pas l'attribut « Sud-Est » (des U.S.). Or, le fait qu'il possède le premier implique la possession du deuxième même si ce n'est pas explicitement signifié dans la source d'où a été extraite la connaissance. Cette relation d'héritage peut donc être utilisée pour compléter la liste des attributs d'un contexte formel.

3.5 Synthèse

Les domaines abordés dans ce chapitre utilisent différents langages de représentation (UML, ER, OWL, etc.) et proposent différentes approches de conversion. Ces conversions s'intéressent, par exemple, à faire de la migration de bases de données, du refactoring de code ou du mapping d'ontologie.

La table 3.1 décrit les articles traités avec leur date de publication, le domaine de recherche en informatique concerné, l'objectif visé, les langages concernés, la prise en compte des données dans la conversion. Par rapport aux descriptions des articles que nous avons faites précédemment, nous ajoutons une indication sur la présence ou l'absence d'une validation. On peut y observer la diversité des langages dans les articles sélectionnés. La diversité des dates de publication montre enfin que la question de la conversion se pose depuis longtemps et reste d'actualité.

Article	Date	Domaine	Objectif	Langages	Niveau instance	Validation
[Trujillo and Luján-Mora, 2003]	2003	BD	migration de base de données	base → base	oui	oui - argumentation
[Skoutas et al., 2009]	2009	BD	migration de base de données	graphe → graphe	non	oui - argumentation
[Noy and Rector, 2006]	2006	BD	transformation ternaire	OWL	non	oui - argumentation
[Jones and Song, 2000]	2000	BD	transformation ternaire	ER	non	oui - argumentation
[Fowler, 2018]	2018	GL	migration et optimisation de code	code → code	non	oui - argumentation
[Sunyé et al., 2001]	2001	GL	optimisation de système	UML → UML	non	oui - OCL & argumentation
[de la Vega et al., 2020]	2020	Data Science	construction de jeu de données	jeu de données relationnel → table de données	oui	oui - argumentation
[Bouihi and Bahaj, 2019]	2019	Ontologie	conversion	UML → OWL	oui	non (présente dans [Bouihi and Bahaj, 2016])
[Calvanese et al., 2021]	2021	Ontologie	extraire une ontologie d'une base de données	tables de données → OWL	oui	oui - argumentation
[Ganter and Wille, 1999]	1999	FCA	bases théoriques	contexte formel $\mathcal{K}$	oui	oui - argumentation
[Neznanov et al., 2013]	2013	FCA	conversion	table de données → contexte formel $\mathcal{K}$	oui	non
[Cellier et al., 2008]	2008	FCA	complétion des attributs	$\mathcal{K} \rightarrow \mathcal{K}$	oui	oui - argumentation

TABLE 3.1 – *Récapitulatif des caractéristiques de chaque article étudié : date de parution, domaine de recherche, objectif, langages concernés, prise en compte des données et présence d'une validation. Les domaines de recherche sont présentés par leur acronyme : BD pour base de données, GL pour génie logiciel, FCA pour Analyse de Concepts Formels.*

L'objet de ce mémoire est de proposer une méthode de conversion de données, par exemple fournies dans un tableur comme dans le cas de Knomana, et dont le modèle de

données est un diagramme de classes UML.
Les tables 3.2 et 3.3 présentent une synthèse de chaque article en regard du besoin de conversion du modèle et des données. Pour structurer ces tableaux, les descripteurs utilisés sont les éléments communs à la plupart des langages à classes, avec la terminologie adoptée par UML.
La table 3.2 présente les éléments principaux à convertir en regard de la cible : la classe, l'attribut, l'association binaire simple, l'instance (d'une classe) et le lien (tuple d'une association).
La table 3.3 fait un focus sur les associations nécessitant une transformation avant la conversion. Les éléments d'intérêt sont le type de l'association, son arité et l'héritage (généralisation/spécialisation).

Méthode	**Classe**	**Attribut**		**Association binaire simple**	**Instance & Lien**
		multi-valué	**booléen**		
[Trujillo and Luján-Mora, 2003]	✓ → table cible	× transformation sans scaling		**NT**	✓ → objet d'une table cible
[Skoutas et al., 2009]	**NT**	× transformation sans scaling		**NT**	–
[Noy and Rector, 2006]	**NT**	–		–	–
[Jones and Song, 2000]	**NT**	–		–	–
[Fowler, 2018]	**NT**	**NT**		**NT**	– au niveau code
[Sunyé et al., 2001]	**NT**	**NT**		**NT**	– non mentionnés
[de la Vega et al., 2020]	✓ → table cible	✓ → colonne de la table		✓ → 1 référence = 1 ensemble de colonnes	✓ → 1 ligne
[Bouihi and Bahaj, 2019]	✓ → OWL Classe	✓ → OWL Datatype Property		✓ → OWL Object Property	✓ → OWL individu
[Calvanese et al., 2021]	✓ → OWL Classe	✓ → OWL Datatype Property		✓ → OWL Object Property	✓ → OWL individu
[Ganter and Wille, 1999]	**NT**	✓ scaling	**NT**	– pas d'association avec FCA	**NT**
[Neznanov et al., 2013]	**NT**	✓ scaling rule	✓ simple rule	– pas d'association	✓ 1 ligne → 1 objet de $\mathcal{K}$
[Cellier et al., 2008]	**NT**	✓ complétés via taxinomie		– pas d'association avec FCA	**NT**
Algorithme Right order	✓ → $\mathcal{K}$	✓ scaling	✓ → attribut de $\mathcal{K}$	✓ → r	✓ 1 instance → 1 objet de $\mathcal{K}$ 1 lien → 1 élément de r

TABLE 3.2 – *Comparaison des méthodes vis-à-vis des éléments UML principaux, ou de leurs équivalents, à convertir. L'abréviation* ***NT*** *signifie que l'élément existe dans le langage source et qu'aucun traitement ne lui est appliqué. Le symbole* ✓ *signifie que l'élément subit une conversion ou une transformation. Le symbole* – *signifie que l'élément n'est pas présent dans le domaine considéré par l'article. Le symbole* × *signifie que l'élément existe et subit une transformation mais qu'elle n'est pas applicable dans notre cas à cause du contexte.*

La table 3.2 montre que parmi les domaines étudiés, seul FCA nécessite l'application d'un *scaling* sur les attributs multi-valués. Les autres domaines doivent certes convertir les attributs pour correspondre au format cible mais ils supportent tous la notion d'attributs multi-valués. Enfin, lors d'une conversion, le domaine du Génie Logiciel a tendance à faire abstraction des données elles-mêmes (les instances) et des liens. En Bases de Données les deux niveaux (schéma et données) sont souvent considérés, même si cela ne ressort pas dans tous les articles présentés.

La table 3.3 montre que les différents domaines ne tiennent pas tous compte des aspects spécifiques des relations, à l'exemple de l'agrégation, la composition et la dépendance qui ne sont considérés que par le Génie Logiciel. Il en va de même pour l'héritage qui n'est mobilisé que par le Génie Logiciel, la science des données (dont la méthode est similaire au Génie Logiciel) et pour les ontologies. Enfin, la transformation d'une relation n-aire en plusieurs relations binaires n'est pas présente en Génie Logiciel car il n'y a pas, à

Méthode	**Nature spécifique (agrégation, composition, dépendance)**	**N-aire**	**Héritage**		
			collapse	**push up/down attribut**	**push up/down association**
[Trujillo and Luján-Mora, 2003]	pas indiqué	–	–	–	–
[Skoutas et al., 2009]	pas indiqué	–	–	–	–
[Noy and Rector, 2006]	–	✓ matérialisation	–	–	–
[Jones and Song, 2000]	–	✓ décomposition	–	–	–
[Fowler, 2018]	**NT**	– pas de notion n-aire dans du code	✓	✓	✓ dans le code : association = attribut
[Sunyé et al., 2001]	**NT**	– UML pour code : pas de notion n-aire	✓	✓	✓
[de la Vega et al., 2020]	–	– pas de notion n-aire avec les références	✓	✓	✓
[Bouihi and Bahaj, 2019]	pas indiqué	~ solution OWL répandue → matérialisation	×	×	×
			conversion différente car natif en OWL		
[Calvanese et al., 2021]	–	– mais représentées par la matérialisation	×	×	×
			conversion différente car natif en OWL		
[Ganter and Wille, 1999]	–	–	–	–	–
[Neznanov et al., 2013]	–	–	–	–	–
[Cellier et al., 2008]	–	–	–	–	–
Algorithme Right order	✓ Agg_{Simple} $Comp_{Simple}$ Dep_{Simple}	✓ Na_{mat} Na_{split}	✓	✓	✓

TABLE 3.3 – *Comparaison des méthodes vis-à-vis des associations UML, ou de leurs équivalents, à transformer. L'abréviation* ***NT*** *signifie que l'élément existe dans le langage source et qu'aucun traitement ne lui est appliqué. Le symbole* ✓ *signifie que l'élément subit une conversion ou une transformation. Le symbole* – *signifie que l'élément n'est pas présent dans le domaine considéré par l'article. Le symbole* × *signifie que l'élément existe et subit une transformation mais qu'elle n'est pas applicable dans notre cas à cause du contexte.*

ce jour, de notion de relation n-aire dans du code source de logiciel. Pour les Bases de Données, les ontologies et la science des données, cette transformation est souvent nécessaire et, même s'il en existe d'autres, la technique de transformation la plus répandue est la matérialisation.

Positionnement par rapport à l'état de l'art Suite à cet état de l'art sur quatre différents domaines proposant des conversions, il s'avère que chacun d'eux contribue à décrire des conversions qui nous intéressent dans notre contexte. Les opérations de conversion des éléments UML principaux (i.e. classes, attributs, associations binaires simple et instances) sont relativement similaires. Par contre, les opérations de conversion sur les associations complexes n'existent pas dans tous les domaines. Les bases de données et les ontologies offrent des conversions importantes comme la matérialisation des relations n-aire. La conversion de l'héritage est propre au Génie Logiciel et à la science des données. Tout ceci montre l'intérêt de disposer d'une synthèse de ces opérations de conversion.
Les différents articles étudiés ne proposent pas non plus de mécanisme définissant un ordre d'application de différentes conversions menant à un résultat cohérent. Enfin, nous n'avons pas trouvé de solution pour modéliser les données indéterminées. Dans cette thèse, nous chercherons à combler ces différents manques.

3.6 Conclusion

Dans ce chapitre, quatre domaines incluant des approches de conversion ont été abordés : les bases de données (section 3.1), le génie logiciel (section 3.2), la science des données (section 3.3) et la représentation des connaissances (section 3.4) où nous avons identifié des approches pour les ontologies (sous-section 3.4.1) et pour l'Analyse de Concepts Formels (sous-section 3.4.2). La synthèse et le positionnement de la suite du mémoire ont été présentés en section 3.5.

Le chapitre suivant présente les méthodes de conversion des données élaborées au cours de la thèse. Nous verrons plus particulièrement comment convertir un jeu de données modélisé par un diagramme de classes UML vers une famille de contextes relationnels.

IV

Méthodes de conversion des connaissances

L'utilisation d'une méthode de découverte de connaissances, par exemple une méthode de classification, telle que l'Analyse de Concepts Formels, nécessite de sélectionner, préparer et convertir les données descriptives des connaissances selon le format accepté par la méthode de classification [Fayyad et al., 1996].

Pour appliquer les méthodes de classification et d'exploration basées sur FCA, les connaissances de Knomana doivent être converties pour correspondre au format de FCA. La structure d'une connaissance étant de type relationnel, le format d'entrée correspond à celui de l'Analyse de Concepts Relationnels, demandant en entrée que les données soient présentées sous la forme d'une famille de contextes relationnels (RCF).

Ce chapitre présente les étapes du processus de conversion des données, dont le modèle est exprimé en UML, vers une famille de contextes relationnels (RCF). Ce processus est décrit en section 4.1. Des cas particuliers de conversion sont ensuite présentés pour modéliser les relations n-aires (section 4.2) et les données indéterminées (section 4.3). Des exemples illustratifs permettant une meilleure compréhension des différentes conversions sont présentés en section 4.4. Enfin la section 4.5 conclut le chapitre en replaçant le processus par rapport aux travaux présentés dans le chapitre 3.

4.1 Conversion depuis un diagramme de classes UML vers une famille de contextes relationnels (RCF)

Nous avons choisi d'utiliser les diagrammes de classes UML pour représenter nos modèles. Ce diagramme UML fournit un large éventail d'outils dont la spécialisation/généralisation (relation également appelée *héritage* dans ce document, héritage étant le nom du

procédé associé à la mise en œuvre de la spécialisation/généralisation), les relations n-aires (appelées *associations* dans le vocabulaire d'UML) et les attributs complexes (non binaires). Cela permet donc de modéliser un domaine avec une large palette de concepts pour la modélisation. Les modèles UML et RCF sont de plus relativement faciles à mettre en correspondance.

En UML [Object Management Group (OMG), 2017], les différents types d'objets (ou instances) sont regroupés dans des classes comportant des attributs de types variés et des opérations. Il peut y avoir plusieurs sortes de relations entre les classes. UML réserve le terme de *relation* pour les relations entre classes qui n'ont pas de contre-partie au niveau des objets (pas de liens entre objets). Par exemple, la relation de spécialisation/généralisation (encore appelée héritage) permet d'organiser hiérarchiquement les classes au moyen de la relation *sorte de* ou *is-a*, à l'exemple de la classe *Carnivore* qui spécialise la classe *Animal*. Mais on ne place pas de spécialisation/généralisation entre deux instances de ces classes. UML propose également des relations de dépendances souvent annotées au moyen d'un stéréotype (par exemple *produit* entre une classe *FabriqueDeBiopesticides* et une classe *Biopesticide*). Le terme d'*association* est utilisé pour les connections entre classes qui ont une contre-partie au niveau des objets (des liens entre objets). Les associations permettent justement de regrouper, de représenter et d'abstraire des liens entre des objets partageant une même sémantique, à l'exemple du lien *consommer* associant un animal et un végétal. Les relations de dépendance ont une sémantique peu précise et, quoiqu'UML ne les considère pas comme des associations, elles sont traitées dans ce document avec ces dernières. Dans le document, quand il n'y a pas d'ambiguité, nous ferons parfois des glissements de vocabulaire entre relation (utilisé dans le sens large), relation (avec le sens restreint d'UML) et association (avec le sens d'UML).

Dans une famille de contextes relationnels (RCF), les différents types d'objets sont représentés avec des contextes formels ayant des attributs booléens. Les liens entre les objets sont regroupés dans des contextes relationnels, chacun correspondant à une association binaire, unidirectionnelle et sans attribut d'UML. Ces contextes relationnels sont en effet de simples relations binaires entre objets de contextes formels. Avec ces simples éléments, le modèle de données d'une RCF peut donc être mis en correspondance avec un diagramme de classes UML restreint (voir définition 4.1.1). On peut ensuite transformer les éléments du diagramme UML (classes, attributs et associations) en éléments d'une RCF (contextes formels et relationnels).

Pour convertir un diagramme de classes UML vers une RCF, il suffit donc de le transformer en un autre diagramme de classes beaucoup plus restreint. Pour cela, nous avons défini une méthode en plusieurs étapes illustrées par la figure 4.1.

La première étape consiste à transformer le diagramme initial pour que les classes et les associations correspondent aux restrictions d'une RCF. Par contre les attributs ne sont pas encore traités. À l'issue de cette première étape, on obtient un diagramme de type UML_{CoBUMu} respectant la définition 4.1.1.

Définition 4.1.1. $\boldsymbol{UML_{CoBUMu}}$ *Un diagramme de type* UML_{CoBUMu} *est un diagramme de classes UML restreint. Il contient uniquement les éléments suivants : classes concrètes, attributs multi-valués et associations. Les associations sont binaires, sans attribut et unidirectionnelles. Il n'y a aucune indication d'agrégation ni de composition.*

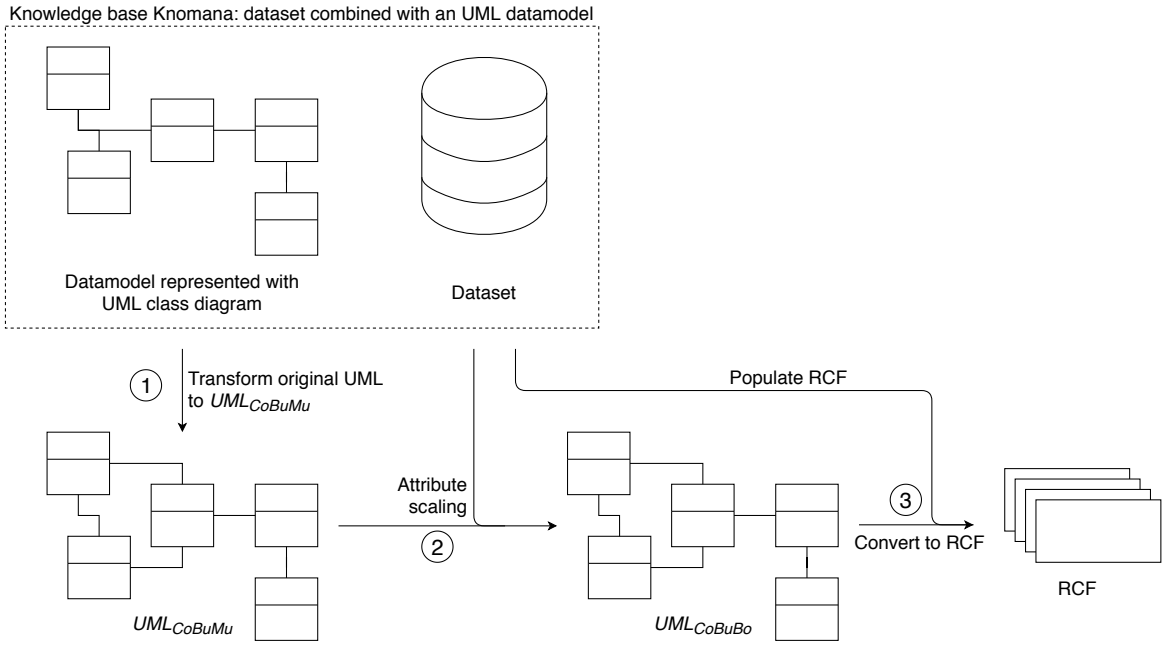


FIGURE 4.1 – *Processus de conversion d'un diagramme de classes UML en une RCF. Étape 1 : Transformer toutes les associations. Étape 2 : Transformer tous les attributs multi-valués en attributs booléens. Étape 3 : Convertir la nouvelle version du diagramme en RCF.*

La deuxième étape consiste à appliquer différentes méthodes de graduation, appelées méthodes *de scaling*, sur les attributs multi-valués afin de transformer chacun d'entre eux en un ensemble d'attributs booléens. À l'issue de cette étape, on obtient un diagramme de type UML_{CoBUBo} respectant la Définition 4.1.2.

Définition 4.1.2. $\boldsymbol{UML_{CoBUBo}}$ *Un diagramme de type* UML_{CoBUBo} *est un diagramme de classes UML ayant les mêmes restrictions qu'un diagramme* UML_{CoBUMu} *(classes concrètes, associations binaires, sans attributs et unidirectionnelles) mais dont les attributs sont en plus uniquement booléens.*

Pour finir, la troisième étape consiste à convertir chaque élément du diagramme UML_{CoBUBo} en un élément d'une RCF et à peupler la RCF avec les données du jeu de données.

Algorithme de conversion entre un diagramme de classes UML et une RCF
Afin d'éviter d'appliquer deux fois une même transformation, notamment lors de la première étape, un algorithme nommé « Right order » a été défini (algorithme 1). Cet algorithme définit l'ordre de priorité des transformations. Pour cela, l'étape 1 est décomposée en deux parties. La première partie applique les transformations les plus complexes, c'est-à-dire celles concernant l'héritage (les relations de spécialisation/généralisation) avec les lignes 1 et 2. En effet, ces transformations ont un impact aussi bien sur les classes que sur les associations. La partie 2 traite ensuite toutes les associations entre les lignes 3 à 11 pour les transformer en associations simples, c'est-à-dire binaires, sans attribut et unidirectionnelles. Lors de l'étape 2, pour obtenir le modèle $\boldsymbol{UML_{CoBUBo}}$, différents scaling sont appliqués sur les attributs multi-valués pour produire des ensembles d'attributs booléens (lignes 12 à 14). Enfin, l'étape 3 permet la conversion en RCF via la transformation

de chaque classe et de chaque association du modèle UML respectivement en contexte formel (lignes 15 à 22) et relationnel (lignes 23 à 26).

Algorithm 1: Right order

Data: original UML class diagram verifying hypotheses: no static attributes, each abstract class `C` has at least a concrete subclass

Result: A RCF preserving instances and their description, triples (instance, attribute, value) and associations (except for solution2) of the original UML class diagram

```
   // First step
   // Part One: Generalization
 1 if there is a specialization/generalization then
 2 |  choose representation level and apply generalization transformations;
   // Part Two: Associations
 3 if there is an association then
 4 |  foreach aggregation or composition or dependency do
 5 |  |  transform it to a simple association;
 6 |  foreach association with attribute do
 7 |  |  materialize the association;
 8 |  foreach n-ary association do
 9 |  |  choose Na_mat or Na_split (last column, row 8 and 9 in Table 4.2) and apply the
   |  |   corresponding transformation;
10 |  foreach bidirectional association do
11 |  |  make two unidirectional associations;
   // Second step: Multi-valued attributes
12 if there is one multi-valued attribute then
13 |  foreach multi-valued attribute do
14 |  |  scale it to convert into a set of boolean attributes;
   // Third step: Conversion into an RCF
   // at this step, the model contains only classes, boolean attributes and
      associations (binary, unidirectional, without attribute and between
      concrete classes)
15 foreach concrete class C do
16 |  create a formal context K, with K's name = C's name;
17 |  foreach attribute of C do
18 |  |  give the same attribute to K;
19 |  foreach instance i of C do
20 |  |  create a line l in the formal context K;
21 |  |  foreach attribute value v do
22 |  |  |  assign v in the line l;
23 foreach association a do
24 |  create a relational context r_a;
25 |  foreach link l of a do
26 |  |  assign the target of l to the source of l in r_a;
27 return the RCF;
```

Les sous-sections suivantes décrivent les transformations appliquées lors de la conversion d'un modèle UML. Chaque transformation y est décrite par rapport au modèle (classes et associations) mais également par rapport aux instances des classes et aux

liens des associations. En effet, les modifications effectuées sur le modèle entraînent des modifications (migrations) au niveau des instances et des liens entre ces instances. Pour chaque transformation, les migrations associées ont été formalisées.

Notation : Durant cette section, une notation spécifique est appliquée pour différencier les éléments. Pour les diagrammes de classes UML, les notations utilisées seront les suivantes : $\mathtt{C}_{NomClasse}$ pour les classes, $\mathtt{r}_{NomRelation}$ pour les associations, $\mathtt{a}_{NomAttribut}$ pour les attributs, $\mathtt{i}_{NomInstance}$ pour les instances et $\mathtt{l}_{NomLien}$ pour les liens. Les exposants $^{\mathtt{M}_1}$ et $^{\mathtt{M}_2}$ signifient respectivement qu'on mentionne un élément avant et après la transformation. Pour la FCA, les notations utilisées seront les suivantes : $\mathcal{K}_{name}$ pour les contextes formels et r_{Name} pour les contextes relationnels.

4.1.1 Étape 1 : Transformation des éléments du diagramme non compatibles

Cette sous-section 4.1.1 présente les différentes transformations appliquées lors de l'étape 1. Chaque élément du diagramme de classes (exceptés les attributs) qui ne respecte pas les contraintes d'une RCF subit une transformation. Pour cela, il faut adapter les relations d'héritage (de spécialisation/généralisation) et les associations.

Transformation de l'héritage

Dans un diagramme de classes UML, et notamment pour la relation d'héritage, les descriptions sont réparties dans et autour des différentes classes. Ces descriptions seront intégrées dans les contextes de la RCF. Pour représenter l'héritage en UML, on utilise une hiérarchie de classes, chaque niveau ayant sa propre description. Dans les sous-classes, les propriétés des super-classes ne sont pas réécrites, puisqu'elles sont héritées. La sémantique de l'héritage veut que les objets des sous-classes disposent des propriétés des super-classes (attributs, méthodes, associations) avec d'éventuels raffinements, lorsque les propriétés sont redéfinies. Une RCF ne contient que des tables binaires qui décrivent soit des objets par leur attributs (contextes formels), soit des liens entre des objets de catégories différentes (contextes relationnels). Il n'y a donc pas de moyen explicite de représenter l'héritage dans une RCF. Une solution serait de représenter la spécialisation par un contexte relationnel que l'on pourrait nommer, par exemple, « $\mathtt{r}_{isA}$ », en prévoyant également un contexte formel décrivant les classes. Néanmoins, cela complique la modélisation car cela demande de modéliser le niveau des classes. En effet, l'héritage n'est pas une relation entre objets mais entre classes.

Pour représenter la notion d'héritage dans la RCF, nous avons donc choisi d'aplatir la hiérarchie. L'idée est de projeter les instances des sous-classes dans leurs super-classes. En effet, si une instance appartient à une classe, elle appartient également à toutes ses super-classes ; c'est le principe de base de l'héritage. Nous exploitons cette caractéristique pour représenter l'héritage dans la RCF.

Pour ce faire, un niveau de représentation doit d'abord être choisi (e.g. par un expert) en sélectionnant des classes de la hiérarchie. Ce niveau permet de réduire ou d'agrandir le nombre de classes (et donc de contextes formels) à traiter ayant un réel intérêt pour l'expert. Les différents éléments de chacune des classes (attributs, associations, instances et liens) sont ensuite projetés dans le niveau sélectionné. Pour choisir un niveau de représentation, il faut, pour chaque classe feuille $\mathtt{C}_F$ de la hiérarchie, sélectionner une classe $\mathtt{C}_{Ch}$. La classe $\mathtt{C}_{Ch}$ peut être une feuille elle-même ou n'importe laquelle des super-classes de la classe feuille, qu'elle soit concrète ou abstraite. De plus, deux classes feuilles $\mathtt{C}_F$ différentes peuvent avoir la même $\mathtt{C}_{Ch}$. Il peut donc y avoir entre 1 et n classes choisies, n étant le nombre de feuilles de la hiérarchie avant la transformation.

La table 4.1 décrit les modifications de modèle à effectuer dans chaque branche de la hiérarchie une fois toutes les classes $\mathtt{C}_{Ch}$ choisies. Si une classe choisie $\mathtt{C}_{Ch}$ était une classe abstraite, elle change de statut et devient une classe concrète. Suivant le principe de l'héritage, les attributs et associations sont propagés dans toutes les sous-classes d'une classe. De plus, les sous-classes de chaque classe $\mathtt{C}_{Ch}$ choisie y étant projetées (ces sous-classes sont comme fusionnées dans $\mathtt{C}_{Ch}$), les classes $\mathtt{C}_{Ch}$ choisies récupèrent également les propriétés de leurs sous-classes ainsi qu'un nouvel attribut booléen par sous-classe pour représenter l'appartenance d'une instance à ladite sous-classe. Enfin, les classes abstraites restantes et les sous-classes de chaque $\mathtt{C}_{Ch}$ sont ensuite supprimées du diagramme.

Element to adapt	**Model modification**
Classes	
abstract superclasses of $\mathtt{C}_{Ch}$ $\{\mathtt{C}_{Super_i} \mid i = 1, ...n_{super}\}$	deleted
concrete superclasses of $\mathtt{C}_{Ch}$ $\{\mathtt{C}_{Super_i} \mid i = 1, ...n_{super}\}$	preserved (subject to modifications on attributes or associations)
chosen class $\mathtt{C}_{Ch}$	preserved and (if abstract) converted into a concrete class (subject to modifications on attributes or associations)
subclass of $\mathtt{C}_{Ch}$ $\{\mathtt{C}_{Sub_i} \mid i = 1, ...n_{sub}\}$	projected on the corresponding $\mathtt{C}_{Ch}$ and deleted
Attributes	
superclasses $\{\mathtt{C}_{Super_i}\}$	own + inherited from own superclasses
chosen class $\mathtt{C}_{Ch}$	own + inherited from $\{\mathtt{C}_{Super_i}\}$ + projected from $\{\mathtt{C}_{Sub_i}\}$
Associations	
superclasses $\{\mathtt{C}_{Super_i}\}$	own + inherited from own superclasses
chosen class $\mathtt{C}_{Ch}$	own + inherited from $\{\mathtt{C}_{Super_i}\}$ + projected from $\{\mathtt{C}_{Sub_i}\}$

TABLE 4.1 – *Heritage : Transformation de modèle à appliquer pour chaque branche de la hiérarchie (i.e. chaque chemin reliant la racine à une feuille). Chaque branche possède une unique* $\mathtt{C}_{Ch}$*. Plusieurs branches peuvent avoir la même* $\mathtt{C}_{Ch}$*. Une classe abstraite peut être choisie en tant que* $\mathtt{C}_{Ch}$*.*

La figure 4.2 illustre, sur un exemple, une transformation appliquée pour convertir l'héritage. Le modèle d'origine comporte 4 classes, dont 3 font partie de la même hiérarchie, et une association. La hiérarchie est composée de la classe principale $\mathtt{C}^{\mathtt{M}_1}_{Document}$, spécialisée par la classe $\mathtt{C}^{\mathtt{M}_1}_{Article}$, elle-même spécialisée par $\mathtt{C}^{\mathtt{M}_1}_{JournalArticle}$. L'association $\mathtt{r}^{\mathtt{M}_1}_{isDescribedIn}$ entre $\mathtt{C}^{\mathtt{M}_1}_{Plant}$ et la hiérarchie indique qu'une plante est décrite dans un document. La classe $\mathtt{C}^{\mathtt{M}_1}_{JournalArticle}$ est l'unique feuille de la hiérarchie. Pour cet exemple, nous avons choisi d'aplatir la hiérarchie au niveau de $\mathtt{C}^{\mathtt{M}_1}_{Article}$ (symbolisé par l'étoile violette).
Les superclasses concrètes de $\mathtt{C}^{\mathtt{M}_1}_{Article}$ (i.e. la classe $\mathtt{C}^{\mathtt{M}_1}_{Document}$) ne subissent pas de mo-

dification. Les classes abstraites, quand il y en a, disparaissent et les associations qui les concernent sont déplacées vers les sous-classes. Toutes les sous-classes de $\mathtt{C}^{\mathtt{M}_1}_{Article}$ (i.e. la classe $\mathtt{C}^{\mathtt{M}_1}_{JournalArticle}$) sont ensuite projetées dans $\mathtt{C}^{\mathtt{M}_1}_{Article}$. La classe $\mathtt{C}^{\mathtt{M}_2}_{Article}$ se retrouve donc avec deux nouveaux attributs supplémentaires : $\mathtt{a}_{journalName}$, l'attribut propre à la classe $\mathtt{C}^{\mathtt{M}_1}_{JournalArticle}$, et $\mathtt{a}_{JournalArticle}$, un nouvel attribut booléen, créé à partir du nom de la classe projetée. Ce nouvel attribut permet d'indiquer si une instance était de la classe $\mathtt{C}^{\mathtt{M}_1}_{JournalArticle}$.
Les relations d'héritage sont ensuite supprimées. Chaque classe de la hiérarchie encore présente récupère les attributs et les associations de l'ensemble de ses super-classes. Ainsi, après la transformation, $\mathtt{C}^{\mathtt{M}_2}_{Article}$ possède les attributs hérités de $\mathtt{C}^{\mathtt{M}_1}_{Document}$ ($\mathtt{a}_{title}$, $\mathtt{a}_{year}$ et $\mathtt{a}_{author}$), ses attributs propres ($\mathtt{a}_{pages}$) et ceux provenant de la projection de $\mathtt{C}^{\mathtt{M}_1}_{JournalArticle}$ ($\mathtt{a}_{journalName}$ et $\mathtt{a}_{journalArticle}$). Après la transformation, le nouveau diagramme UML possède donc trois classes : $\mathtt{C}^{\mathtt{M}_2}_{Plant}$ (n'ayant subi aucune modification), $\mathtt{C}^{\mathtt{M}_2}_{Document}$ et $\mathtt{C}^{\mathtt{M}_2}_{Article}$. L'association entre $\mathtt{C}^{\mathtt{M}_2}_{Plant}$ et $\mathtt{C}^{\mathtt{M}_2}_{Document}$ est dupliquée entre $\mathtt{C}^{\mathtt{M}_2}_{Plant}$ et $\mathtt{C}^{\mathtt{M}_2}_{Article}$ car c'est une association héritée dans le modèle original.

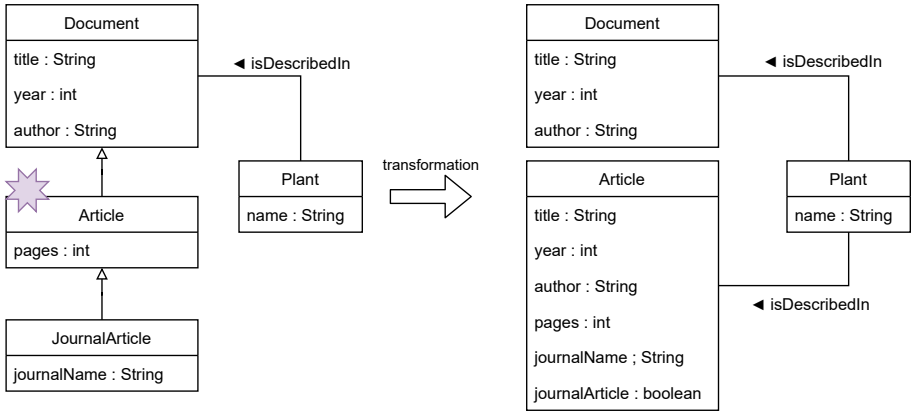
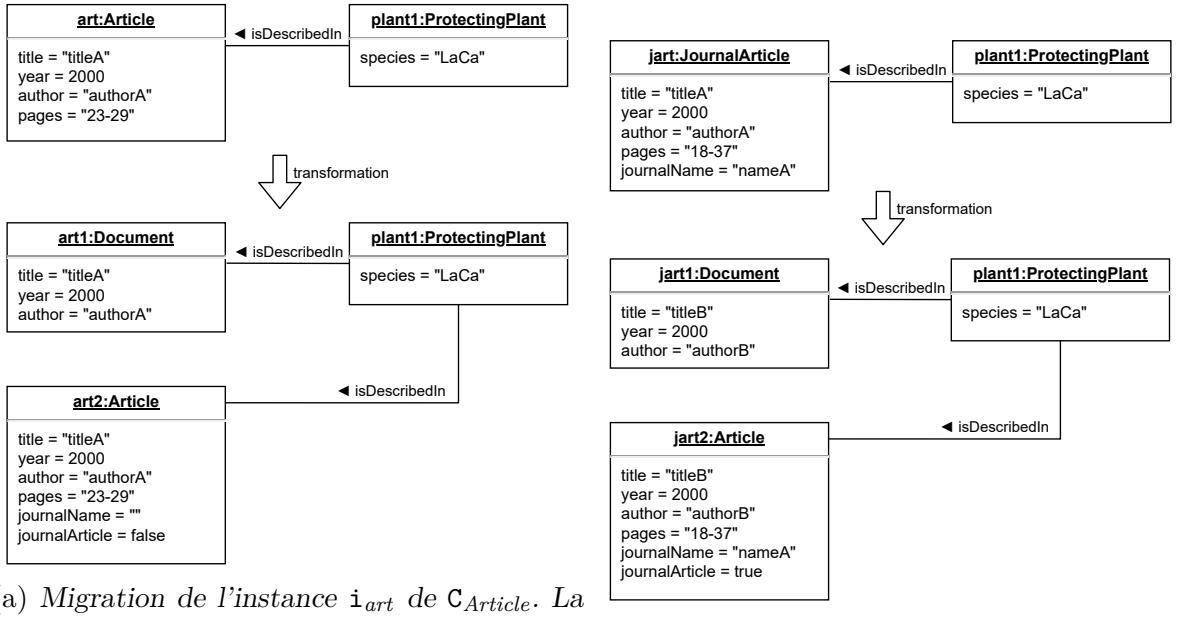


FIGURE 4.2 – *Exemple de transformation d'un modèle UML contenant de l'héritage. On considère qu'un expert a choisi* $\mathtt{C}_{Article}$ *comme niveau de représentation (symbolisé par l'étoile violette).* $\mathtt{C}_{JournalArticle}$ *est donc projetée (fusionnée) dans* $\mathtt{C}_{Article}$ *ce qui ajoute deux attributs à* $\mathtt{C}_{Article}$. $\mathtt{C}_{Document}$ *est inchangée et les relations d'héritage disparaissent. L'association* $\mathtt{r}_{isDescribedIn}$ *entre* $\mathtt{C}_{Plant}$ *et* $\mathtt{C}_{Document}$ *est héritée par* $\mathtt{C}_{Article}$ *(dans la hiérarchie de gauche), elle est donc ajoutée par duplication.*

Les deux schémas de la figure 4.3 illustrent la migration de deux instances opérée lors de la transformation. La sous-figure 4.3a montre la migration d'une instance de $\mathtt{C}_{Article}$. Pour garder la notion d'héritage au niveau des instances, les articles étant également des documents, l'instance $\mathtt{i}_{art}$ de $\mathtt{C}^{\mathtt{M}_1}_{Article}$ est dupliquée dans $\mathtt{C}^{\mathtt{M}_2}_{Document}$ ($\mathtt{i}_{art1}$) en plus d'être présente dans $\mathtt{C}^{\mathtt{M}_2}_{Article}$ ($\mathtt{i}_{art2}$). L'instance $\mathtt{i}_{art1}$ possède les attributs de $\mathtt{C}^{\mathtt{M}_2}_{Document}$ alors que l'instance $\mathtt{i}_{art2}$ possède ceux de $\mathtt{C}^{\mathtt{M}_2}_{Article}$. Les valeurs pour les deux attributs ajoutés lors de la projection sont des valeurs par défaut, i.e. `vide` pour $\mathtt{a}_{journalName}$ et `false` pour $\mathtt{a}_{journalArticle}$.
La sous-figure 4.3b montre la migration d'une instance de $\mathtt{C}_{JournalArticle}$. $\mathtt{C}^{\mathtt{M}_1}_{JournalArticle}$ ayant été projetée dans $\mathtt{C}^{\mathtt{M}_2}_{Article}$, l'instance $\mathtt{i}_{jart}$ s'y retrouve donc également projetée et devient l'instance $\mathtt{i}_{jart2}$ de $\mathtt{C}^{\mathtt{M}_2}_{Article}$. De même que pour $\mathtt{i}_{art1}$, l'instance $\mathtt{i}_{jart}$ est également dupliquée dans $\mathtt{C}^{\mathtt{M}_2}_{Document}$ ($\mathtt{i}_{jart1}$). Les attributs de $\mathtt{i}_{jart1}$ et $\mathtt{i}_{jart2}$ sont les mêmes que pour $\mathtt{i}_{art1}$ et $\mathtt{i}_{art2}$, seules les valeurs changent. Notamment, la valeur de l'attribut $\mathtt{a}_{journalName}$ est reprise de l'instance $\mathtt{i}_{jart}$ (i.e. `nameA`) et l'attribut $\mathtt{a}_{journalArticle}$ est associé à `true` car $\mathtt{i}_{jart}$ était

une instance de $\mathtt{C}^{\mathtt{M}_1}_{JournalArticle}$.
Des explications plus formelles concernant la migration d'instance pour l'héritage sont fournies dans Migration 1.



(a) *Migration de l'instance* $\mathtt{i}_{art}$ *de* $\mathtt{C}_{Article}$. *La transformation a pour effet de dupliquer* $\mathtt{i}_{art}$ *dans* $\mathtt{C}^{\mathtt{M}_2}_{Document}$ ($\mathtt{i}_{art1}$) *et* $\mathtt{C}^{\mathtt{M}_2}_{Article}$ ($\mathtt{i}_{art2}$). *L'instance* $\mathtt{i}_{art1}$ *ne possède que les attributs de* $\mathtt{C}^{\mathtt{M}_2}_{Document}$. *L'instance* $\mathtt{i}_{art2}$ *possède les mêmes attributs que* $\mathtt{i}_{art}$ *ainsi les deux nouveaux attributs contenus dans* $\mathtt{C}^{\mathtt{M}_2}_{Article}$. *Le premier* $\mathtt{a}_{journalName}$ *est vide et la valeur* `false` *est affectée à* $\mathtt{a}_{journalArticle}$ *car* $\mathtt{i}_{art}$ *n'est pas un article de journal.*

(b) *Migration de l'instance* $\mathtt{i}_{jart}$ *de* $\mathtt{C}_{JournalArticle}$. *La transformation a pour effet de dupliquer* $\mathtt{i}_{jart}$ *dans* $\mathtt{C}^{\mathtt{M}_2}_{Document}$ ($\mathtt{i}_{jart1}$) *et* $\mathtt{C}^{\mathtt{M}_2}_{Article}$ ($\mathtt{i}_{jart2}$). *L'instance* $\mathtt{i}_{jart1}$ *ne possède les mêmes attributs que* $\mathtt{i}_{jart}$. *La valeur* `true` *est affectée à l"attribut* $\mathtt{a}_{journalArticle}$ *de journal.*

FIGURE 4.3 – *Illustration de la migration de deux instances lors de la transformation de l'héritage présentée en figure 4.2. Durant la transformation,* $\mathtt{C}_{Article}$ *et* $\mathtt{C}_{JournalArticle}$ *sont fusionnées. Chacune des deux sous-figures représente une instance de ces classes.*

Migration 1. *Soient* $\mathtt{M}_1$ *et* $\mathtt{M}_2$ *les diagrammes de classes avant et après la transformation. Soit* $\mathtt{C}_{Ch}$ *la classe choisie pour une branche,* $\mathtt{C}_{Sup}$ *chaque super-classe de* $\mathtt{C}_{Ch}$ *et* $\mathtt{C}_{Sub}$ *chaque sous-classe* $\mathtt{C}_{Ch}$.

- *Les valeurs des attributs de* $\mathtt{i}_{\mathtt{C}}^{\mathtt{M}_1}$ *sont conservées dans* $\mathtt{i}_{\mathtt{C}}^{\mathtt{M}_2}$.
- *Pour chaque lien* $\mathtt{l}^{\mathtt{M}_1}$ *ayant comme source ou destination une instance* $\mathtt{i}^{\mathtt{M}_1}$, *il y a pour chacune des instances* $\mathtt{i}^{\mathtt{M}_2}$ *correspondantes un lien correspondant* $\mathtt{l}^{\mathtt{M}_2}$ *par la migration.*
- ***Migration pour chaque*** $\mathtt{C}^{\mathtt{M}_1}_{Sup}$***:***
  - *Pour chaque instance* $\mathtt{i}^{\mathtt{M}_1}_{\mathtt{C}_{Sup}}$ *de* $\mathtt{C}^{\mathtt{M}_1}_{Sup}$ *il y a dans* $\mathtt{C}^{\mathtt{M}_2}_{Sup}$ *une instance correspondante* $\mathtt{i}^{\mathtt{M}_2}_{\mathtt{C}_{Sup}}$ *par la migration.*
- ***Migration pour chaque*** $\mathtt{C}^{\mathtt{M}_1}_{Ch}$***:***
  - *Pour chaque instance* $\mathtt{i}^{\mathtt{M}_1}_{\mathtt{C}_{Ch}}$ *de* $\mathtt{C}^{\mathtt{M}_1}_{Ch}$ *il y a dans* $\mathtt{C}^{\mathtt{M}_2}_{Sup}$ *une instance correspondante* $\mathtt{i}^{\mathtt{M}_2}_{\mathtt{C}_{Sup}}$ *par la migration.*
  - *Pour chaque* $\mathtt{i}^{\mathtt{M}_2}_{\mathtt{C}_{Sup}}$ *correspondante, les attributs sont ceux de* $\mathtt{C}^{\mathtt{M}_2}_{Sup}$.

  - *Pour chaque* $\mathtt{i}_{C_{Ch}}^{M_1}$ *il y a dans* $\mathtt{C}_{Ch}^{M_2}$ *une instance correspondante* $\mathtt{i}_{C_{Ch}}^{M_2}$ *par la migration.*
  - *Pour chaque* $\mathtt{i}_{C_{Ch}}^{M_2}$ *correspondante, les valeurs des attributs migrés depuis* $\mathtt{C}_{Sub}^{M_1}$ *ont des valeurs par défaut.*
    L'attribut booléen représentant l'appartenance à une classe $\mathtt{C}_{Sub}^{M_1}$ *a pour valeur* `false`*.*
- ***Migration pour chaque*** $\mathtt{C}_{Sub}^{M_1}$***:***
  - *Pour chaque instance* $\mathtt{i}_{C_{Sub}}^{M_1}$ *de* $\mathtt{C}_{Sub}^{M_1}$ *il y a dans* $\mathtt{C}_{Sup}^{M_2}$ *une instance correspondante* $\mathtt{i}_{C_{Sup}}^{M_2}$ *par la migration.*
  - *Pour chaque* $\mathtt{i}_{C_{Sup}}^{M_2}$ *correspondante, les attributs sont ceux de* $\mathtt{C}_{Sup}^{M_2}$*.*
  - *Pour chaque* $\mathtt{i}_{C_{Sub}}^{M_1}$ *il y a dans* $\mathtt{C}_{Ch}^{M_2}$ *une instance correspondante* $\mathtt{i}_{C_{Ch}}^{M_2}$ *par la migration.*
  - *Pour chaque* $\mathtt{i}_{C_{Ch}}^{M_2}$ *correspondante, les attributs sont ceux de* $\mathtt{C}_{Ch}^{M_2}$*.*
  - *Pour chaque* $\mathtt{i}_{C_{Ch}}^{M_2}$ *correspondante, les attributs valués migrés depuis* $\mathtt{C}_{Sub}^{M_1}$ *prennent les valeurs de* $\mathtt{i}_{C_{Sub}}^{M_1}$*.*
    L'attribut booléen représentant l'appartenance à une classe $\mathtt{C}_{Sub}^{M_1}$ *prend pour valeur* `true`*.*

Héritage multiple Les explications données ci-dessus sont appliquées à une transformation concernant l'héritage simple, i.e. pour un diagramme de classes UML où chaque classe ne peut avoir qu'une seule super-classe par niveau. Pour appliquer la transformation à des situations d'héritage multiple, il faut suivre le même principe en ayant une attention particulière aux éléments de même nom. En effet, si deux éléments de même type possèdent le même nom, leurs significations peuvent être différentes. Pour conserver les deux, on peut simplement renommer l'un des deux.

Transformation des associations complexes

La table 4.2 explique les opérations atomiques à appliquer pour transformer une association complexe `r` en une association respectant les contraintes d'une RCF. Il faut prendre en compte l'arité, la présence d'attributs sur l'association, la nature des éléments mis en association et la navigabilité.

Deux transformations de relation n-aire seront détaillées : la matérialisation ($\mathrm{Na}_{\mathrm{mat}}$) qui crée une nouvelle classe pour représenter la relation et le *split* ($\mathrm{Na}_{\mathrm{split}}$) qui crée un graphe complet avec toutes les classes concernées. En UML, l'ajout d'un attribut sur une association revient à faire une classe d'association. La transformation $\mathrm{Att}_{\mathrm{mat}}$ va donc matérialiser cette classe d'association. $\mathrm{Ab}_{\mathrm{Child}}$ fait descendre l'association d'une classe abstraite vers chacune de ses sous-classes. $\mathrm{Bi}_{\mathrm{2uni}}$ produit deux associations inverses à partir d'une association bidirectionnelle. Et enfin, $\mathrm{Agg}_{\mathrm{Simple}}$, $\mathrm{Comp}_{\mathrm{Simple}}$ et $\mathrm{Dep}_{\mathrm{Simple}}$ transforment les agrégations, compositions et dépendances en des associations simples.

Transformation des associations d'agrégation, de composition et de dépendance (resp. $\mathrm{Agg}_{\mathrm{Simple}}$, $\mathrm{Comp}_{\mathrm{Simple}}$ et $\mathrm{Dep}_{\mathrm{Simple}}$) Aucun de ces types d'association

Element to adapt		**ID**	**Description of the transformation**
arity	binary	–	N.A.
	n-ary	Na_{mat}	solution 1 → "materialized" with 1 new class C and n associations (n = number of ends)
		Na_{split}	solution 2 → "split" by using a complete graph between the ends of the association (i.e., $n(n-1)/2$ associations for bidirectional associations)
attribute	with	Att_{mat}	→ "materialized" with 1 new class C and n associations (n = number of ends)
	without	–	N.A.
ends	concrete classes	–	N.A.
	abstract classes	Ab_{Child}	→ move down the association to each child of the class
navigability	unidirectional	–	N.A.
	bidirectional	Bi_{2uni}	binary → make 2 unidirectional associations, one for each direction $\texttt{C}_1 \rightarrow \texttt{C}_2$ and $\texttt{C}_2 \rightarrow \texttt{C}_1$
		–	n-ary → N.A.
	forbidden	$\text{Forb}_{\text{Keep}}$	→ keep the ban
aggregation		$\text{Agg}_{\text{Simple}}$	→ becomes a simple association
composition		$\text{Comp}_{\text{Simple}}$	→ becomes a simple association
dependency		$\text{Dep}_{\text{Simple}}$	→ becomes a simple association

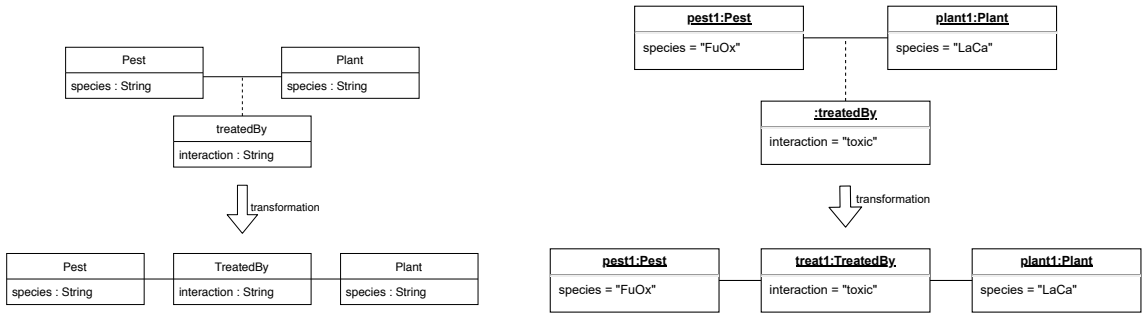
TABLE 4.2 – *Transformations à appliquer sur les associations d'un diagramme de classes UML pour respecter les contraintes d'une RCF. N.A. (Not applicable) signifie qu'aucune transformation n'est requise.*

n'est considéré. Elles sont transformées en une association standard.

Transformation d'une classe d'association (Att_{mat}) Les classes d'association sont matérialisées en utilisant la transformation Att_{mat}, illustrée par la figure 4.4a. Cette transformation produit donc une nouvelle classe ayant pour nom celui de la classe d'association et les attributs de celle-ci restent inchangés. Cette nouvelle classe est ensuite reliée aux extrémités de l'association par de nouvelles associations sans attribut. Si la classe d'association est bidirectionnelle, les associations créées le seront également. Elles devront donc ensuite être transformées en utilisant Bi_{2uni} sur chacune d'entre elles. Sur la figure 4.4a, la classe d'association $\texttt{C}_{\textit{treatedBy}}$ est matérialisée par la nouvelle classe $\texttt{C}_{\textit{TreatedBy}}$. Cette nouvelle classe possède le même attribut que la classe d'association et elle est reliée aux classes $\texttt{C}_{\textit{Pest}}$ et $\texttt{C}_{\textit{Plant}}$ grâce à deux nouvelles associations.

La figure 4.4b illustre la migration opérée lors de la transformation. Les instances $\texttt{i}_{\textit{pest1}}$ et $\texttt{i}_{\textit{plant1}}$ restent les mêmes. L'instance de la classe d'association devient quant à elle l'instance $\texttt{i}_{\textit{treat1}}$ de la nouvelle classe $\texttt{C}_{\textit{TreatedBy}}$ et elle garde ses valeurs d'attributs. De nouveaux liens sont ensuite créés pour relier $\texttt{i}_{\textit{pest1}}$ et $\texttt{i}_{\textit{plant1}}$ à la nouvelle instance $\texttt{i}_{\textit{treat1}}$.

Migration 2 : Pour une classe d'association (Att_{mat}). *Soient* $\texttt{M}_1$ *et* $\texttt{M}_2$ *les diagrammes de classes avant et après la transformation. Selon le méta-modèle UML une classe d'association est à la fois une classe et une association. Soient* $\texttt{C}_1^{\texttt{M}_1}$ *et* $\texttt{C}_2^{\texttt{M}_1}$ *deux classes de* $\texttt{M}_1$*. Soit* $\texttt{A}^{\texttt{M}_1}$ *la classe d'association (attachée visuellement par le trait pointillé et nommée* $\texttt{r}^{\texttt{M}_1}$ *en tant qu'association).* $\texttt{A}^{\texttt{M}_1}$ *a (en tant que classe) un attribut et connecte (en tant qu'association) les classes* $\texttt{C}_1^{\texttt{M}_1}$ *et* $\texttt{C}_2^{\texttt{M}_1}$*.*



(a) *Att_{mat}: Matérialisation de la classe d'association $\mathtt{C}_{treatedBy}$ dans une classe $\mathtt{C}_{TreatedBy}$ connectée à $\mathtt{C}_{Pest}$ et $\mathtt{C}_{Plant}$ avec deux associations. Dans ce qui suit, les valeurs de l'attribut $\mathtt{a}_{interaction}$ incluent* `toxic` *ou* `repulsive`*.*

(b) *Att_{mat}: L'instance de la classe d'association (de type* `treatedBy`*) est transformée en l'instance $\mathtt{i}_{treat1}$ (de type $\mathtt{C}_{TreatedBy}$). Le lien bidirectionnel entre les instances $\mathtt{i}_{pest1}$ et $\mathtt{i}_{plant1}$ est divisé en deux liens bidirectionnels : un entre $\mathtt{i}_{pest1}$ et $\mathtt{i}_{treat1}$; l'autre entre $\mathtt{i}_{treat1}$ et $\mathtt{i}_{plant1}$.*

FIGURE 4.4 – *Att_{mat}: transformation du modèle (à gauche) et migration des instances (à droite).*

- *Pour chaque $\mathtt{i}_{\mathtt{C}_1}^{\mathtt{M}_1}$ instance de $\mathtt{C}_1^{\mathtt{M}_1}$ (resp. $\mathtt{i}_{\mathtt{C}_2}^{\mathtt{M}_1}$ instance de $\mathtt{C}_2^{\mathtt{M}_1}$) dans $\mathtt{M}_1$, il y a dans $\mathtt{C}_1^{\mathtt{M}_2}$ (resp. $\mathtt{C}_2^{\mathtt{M}_2}$) une instance correspondante $\mathtt{i}_{\mathtt{C}_1}^{\mathtt{M}_2}$ (resp. $\mathtt{i}_{\mathtt{C}_2}^{\mathtt{M}_2}$) par la migration.*
- *Les valeurs des attributs sont préservées entre $\mathtt{i}_{\mathtt{C}}^{\mathtt{M}_1}$ et $\mathtt{i}_{\mathtt{C}}^{\mathtt{M}_2}$.*
- *Un lien $\mathtt{l}^{\mathtt{M}_1}$ de $\mathtt{r}^{\mathtt{M}_1}$ entre $\mathtt{i}_{\mathtt{C}_1}^{\mathtt{M}_1}$ et $\mathtt{i}_{\mathtt{C}_2}^{\mathtt{M}_1}$ est transformé en deux liens dans $\mathtt{M}_2$. Le premier est un lien $\mathtt{l}_0^{\mathtt{M}_2}$ entre $\mathtt{i}_{\mathtt{C}_1}^{\mathtt{M}_2}$ et $\mathtt{i}_{\mathtt{C}_\mathtt{A}}^{\mathtt{M}_2}$ (l'instance de la classe $\mathtt{C}_\mathtt{A}^{\mathtt{M}_2}$ générée à partir de $\mathtt{A}^{\mathtt{M}_1}$ par la transformation). Le second est un lien $\mathtt{l}_1^{\mathtt{M}_2}$ entre $\mathtt{i}_{\mathtt{C}_\mathtt{A}}^{\mathtt{M}_2}$ et $\mathtt{i}_{\mathtt{C}_2}^{\mathtt{M}_2}$.*

Transformation d'une association N-aire ($\mathrm{Na}_{\mathrm{mat}}$ et $\mathrm{Na}_{\mathrm{split}}$) La figure 4.5 illustre les transformations d'une relation n-aire avec $\mathrm{Na}_{\mathrm{mat}}$ (à gauche) et $\mathrm{Na}_{\mathrm{split}}$ (à droite). La première ($\mathrm{Na}_{\mathrm{mat}}$) matérialise la relation dans une nouvelle classe et relie cette classe grâce à de nouvelles associations (comme pour la classe d'association avec $\mathrm{Att}_{\mathrm{mat}}$). L'association $\mathtt{r}_{protectionSystem}$ est ainsi matérialisée en la classe $\mathtt{C}_{ProtectionSystem}$ puis est reliée aux trois autres classes. La deuxième ($\mathrm{Na}_{\mathrm{split}}$) casse la relation n-aire et produit un graphe complet entre toutes les classes existantes. Contrairement à la première, la sémantique de la relation n-aire est donc perdue au profit de la génération d'hypothèses. Ainsi, chacune des classes $\mathtt{C}_{Protectingplant}$, $\mathtt{C}_{TargetedOrganism}$ et $\mathtt{C}_{ProtectedOrganism}$ est directement mise en relation avec les deux autres.

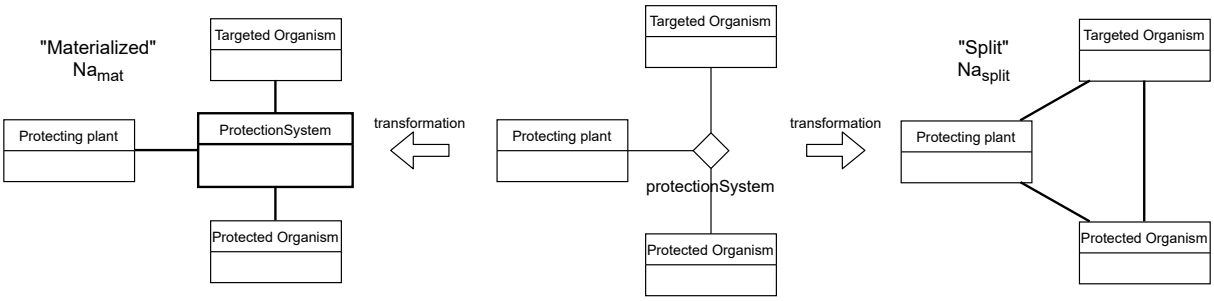


FIGURE 4.5 – *Na_{mat} et Na_{split}: Transformation du modèle. L'association n-aire $\mathtt{r}_{protectionSystem}$ (au milieu) peut être matérialisée dans une classe supplémentaire $\mathtt{C}_{ProtectionSystem}$ en utilisant Na_{mat} (à gauche) ou divisée en un graphe complet de relations binaires avec Na_{split} (à droite).*

La figure 4.6 illustre la transformation d'instance d'une association n-aire par $\mathrm{Na}_{\mathrm{mat}}$ et

Na_{split}, respectivement représentées en vert et rouge. Les instances $\mathtt{i}_{plant1}$, $\mathtt{i}_{prot1}$ et $\mathtt{i}_{target1}$ sont inchangées entre $\mathtt{M1}$ et $\mathtt{M2}$ que ce soit avec Na_{mat} ou Na_{split}. Pour Na_{mat}, une nouvelle instance $\mathtt{i}_{protection1}$ est créée pour la nouvelle classe $\mathtt{C}^{\mathtt{M}_2}_{ProtectionSystem}$. Le lien n-aire $\mathtt{l}^{\mathtt{M}_1}_{protectionSystem}$ est matérialisé par l'instance $\mathtt{i}^{\mathtt{M}_2}_{protection1}$ et trois nouveaux liens sont créés $\mathtt{l}_{uses}$, $\mathtt{l}_{targets}$ et $\mathtt{l}_{protects}$ pour relier respectivement $\mathtt{i}^{\mathtt{M}_2}_{plant1}$, $\mathtt{i}^{\mathtt{M}_2}_{target1}$ et $\mathtt{i}^{\mathtt{M}_2}_{prot1}$ à $\mathtt{i}_{protection1}$. Pour Na_{split}, le lien $\mathtt{l}^{\mathtt{M}_1}_{protectionSystem}$ est divisé en trois liens $\mathtt{l}_{treats}$, $\mathtt{l}_{attacks}$ et $\mathtt{l}_{protects}$ reliant respectivement $\mathtt{i}_{plant1}$ avec $\mathtt{i}_{target1}$, $\mathtt{i}_{target1}$ avec $\mathtt{i}_{prot1}$ et $\mathtt{i}_{plant1}$ avec $\mathtt{i}_{prot1}$.

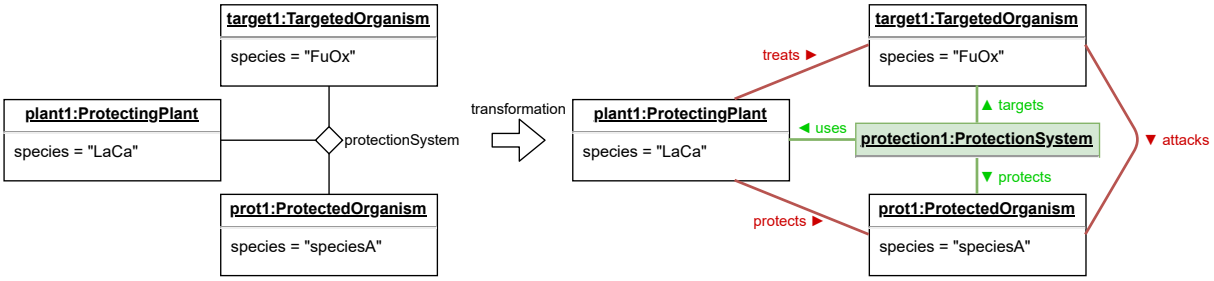


FIGURE 4.6 – *Na_{mat} et Na_{split}: Migration des instances. La partie droite superpose graphiquement les migrations pour Na_{mat}(en vert) et Na_{split}(en rouge). Les éléments noirs sont utilisés dans les deux migrations.*

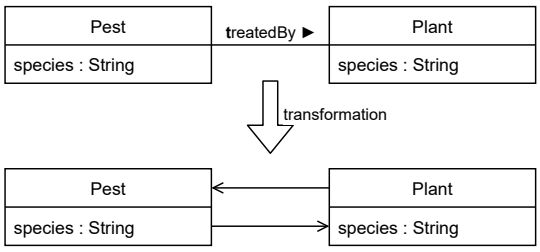
Migration 3 : Pour une association n-aire (Na_{mat} et Na_{split}). *Soient $\mathtt{M}_1$ et $\mathtt{M}_2$ les diagrammes de classes avant et après la transformation. Soient $\mathtt{C}^{\mathtt{M}_1}_n$, $n \in \mathbb{N}$, n classes de $\mathtt{M}_1$. Soit $\mathtt{r}^{\mathtt{M}_1}$ une association n-aire entre les n classes. Soit $\mathtt{C}^{\mathtt{M}_2}_{\mathtt{A}}$ la classe générée à partir de $\mathtt{r}^{\mathtt{M}_1}$ par la transformation Na_{mat}.*

- *Pour chaque instance $\mathtt{i}^{\mathtt{M}_1}_{\mathtt{C}_k}$ de $\mathtt{C}^{\mathtt{M}_1}_k$, $1 \leqslant k \leqslant n$, il y a dans $\mathtt{C}^{\mathtt{M}_2}_k$ une instance correspondante $\mathtt{i}^{\mathtt{M}_2}_{\mathtt{C}_k}$ par la migration.*
- *Les valeurs des attributs sont préservées entre $\mathtt{i}^{\mathtt{M}_1}_{\mathtt{C}}$ et $\mathtt{i}^{\mathtt{M}_2}_{\mathtt{C}}$.*
- ***Pour Na_{mat}:** Le lien n-aire de $\mathtt{r}^{\mathtt{M}_1}$ est transformé en n liens, chaque lien étant entre $\mathtt{i}^{\mathtt{M}_2}_{\mathtt{C}_{\mathtt{A}}}$, l'instance de $\mathtt{C}^{\mathtt{M}_2}_{\mathtt{A}}$, et $\mathtt{i}^{\mathtt{M}_2}_{\mathtt{C}_k}$, $1 \leqslant k \leqslant n$.*
- ***Pour Na_{split}:** Le lien n-aire de $\mathtt{r}^{\mathtt{M}_1}$ est transformé en $(n(n-1))/2$ liens entre $\mathtt{i}^{\mathtt{M}_2}_{\mathtt{C}_k}$, $1 \leqslant k \leqslant n$, et $\mathtt{i}^{\mathtt{M}_2}_{\mathtt{C}_j}$, $1 \leqslant j \leqslant n$, $k \neq j$. Cela forme un graphe complet.*

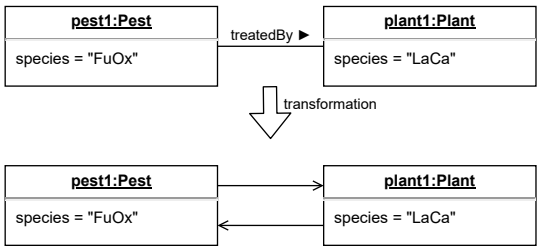
Transformation d'une association bidirectionnelle en deux associations unidirectionnelles (Bi_{2uni}) Pour transformer une association bidirectionnelle, nous appliquons la transformation Bi_{2uni}, illustrée par la figure 4.7a. Cette transformation produit deux associations unidirectionnelles ayant des directions inverses. Ainsi, l'association bidirectionnelle $\mathtt{r}_{treatedBy}$ est décomposée en deux associations unidirectionnelles allant de $\mathtt{C}_{Plant}$ à $\mathtt{C}_{Pest}$ et inversement. Cette transformation n'a pas d'impact sur les classes ni sur leurs attributs.

La figure 4.7b illustre la migration opérée lors de la transformation. Les classes n'ayant pas été modifiées, les instances $\mathtt{i}_{pest1}$ et $\mathtt{i}_{plant1}$ ne changent pas. En revanche, l'association ayant été transformée en deux associations unidirectionnelles inverses, le lien entre les deux instances subit la même transformation.

Migration 4 : Pour une association bidirectionnelle (Bi_{2uni}). *Soient $\mathtt{M}_1$ et $\mathtt{M}_2$ les diagrammes de classes avant et après la transformation. Soient $\mathtt{C}^{\mathtt{M}_1}_1$ et $\mathtt{C}^{\mathtt{M}_1}_2$, deux classes de*



(a) *Bi_{2uni}: Transformation du modèle. Convertit l'association bidirectionnelle $\mathtt{r}_{treatedBy}$ en deux associations unidirectionnelles.*



(b) *Bi_{2uni}: Migration des instances. Le lien bidirectionnel $\mathtt{l}^{\mathtt{M1}}_{treatedBy}$ entre $\mathtt{i}_{pest1}$ et $\mathtt{i}_{plant1}$ est divisé en deux liens unidirectionnels dans M2 : un premier allant de $\mathtt{i}_{pest1}$ à $\mathtt{i}_{plant1}$ et un deuxième allant de $\mathtt{i}_{plant1}$ à $\mathtt{i}_{pest1}$*

FIGURE 4.7 – *Bi_{2uni}: transformation du modèle (à gauche) et migration des instances (à droite).*

$\mathtt{M}_1$ *connectées par une association bidirectionnelle* $\mathtt{r}^{\mathtt{M}_1}$*. Par la transformation, les classes deviennent respectivement* $\mathtt{C}_1^{\mathtt{M}_2}$ *et* $\mathtt{C}_2^{\mathtt{M}_2}$ *et l'association devient* $\mathtt{r}_0^{\mathtt{M}_2}$ *et* $\mathtt{r}_1^{\mathtt{M}_2}$*.*

- *Pour chaque* $\mathtt{i}_{\mathtt{C}_1}^{\mathtt{M}_1}$ *instance de* $\mathtt{C}_1^{\mathtt{M}_1}$ *(resp.* $\mathtt{i}_{\mathtt{C}_2}^{\mathtt{M}_1}$ *instance de* $\mathtt{C}_2^{\mathtt{M}_1}$*) dans* $\mathtt{M}_1$*, il y a dans* $\mathtt{C}_1^{\mathtt{M}_2}$ *(resp.* $\mathtt{C}_2^{\mathtt{M}_2}$*) une instance correspondante* $\mathtt{i}_{\mathtt{C}_1}^{\mathtt{M}_2}$ *(resp.* $\mathtt{i}_{\mathtt{C}_2}^{\mathtt{M}_2}$*) par la migration.*
- *Les valeurs des attributs sont préservées entre* $\mathtt{i}_{\mathtt{C}}^{\mathtt{M}_1}$ *et* $\mathtt{i}_{\mathtt{C}}^{\mathtt{M}_2}$*.*
- *Un lien* $\mathtt{l}^{\mathtt{M}_1}$ *de* $\mathtt{r}^{\mathtt{M}_1}$ *entre* $\mathtt{i}_{\mathtt{C}_1}^{\mathtt{M}_1}$ *et* $\mathtt{i}_{\mathtt{C}_2}^{\mathtt{M}_1}$ *est transformé en deux liens inverses* $\mathtt{l}_0^{\mathtt{M}_2}$ *et* $\mathtt{l}_1^{\mathtt{M}_2}$ *entre* $\mathtt{i}_{\mathtt{C}_1}^{\mathtt{M}_2}$ *et* $\mathtt{i}_{\mathtt{C}_2}^{\mathtt{M}_2}$*.*

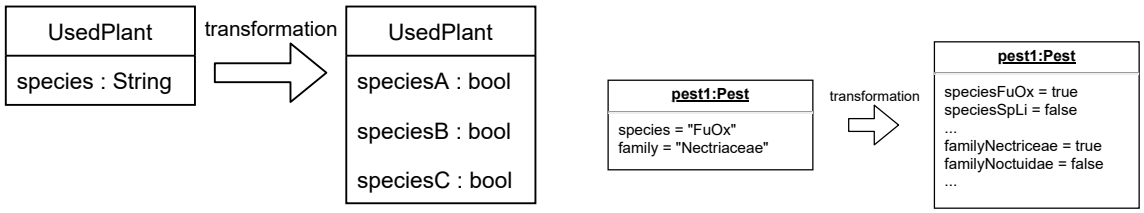
4.1.2 Étape 2 : Génération des attributs booléens (via *scaling* des attributs multi-valués)

RCA, tout comme FCA est une méthode qui n'utilise que des attributs binaires. Un objet possède ou ne possède pas un attribut. En UML, ce genre d'attributs peut être représenté par le type booléen. Lors de cette seconde étape, chaque attribut multi-valué doit donc être transformé en un ensemble d'attributs booléens. Pour cela, différentes méthodes de scaling, inspirées de celles proposées par les références citées dans l'état de l'art, sont appliquées en utilisant les valeurs possibles des attributs.

La figure 4.8 illustre un scaling de type nominal sur un attribut multi-valué. Cet attribut $\mathtt{a}_{species}$ sert à indiquer l'espèce d'une plante ($\mathtt{C}_{UsedPlant}$) et est de type chaîne de caractères (ou String). Ayant autant de valeurs possibles pour $\mathtt{a}_{species}$ que de plantes différentes dans un jeu de données, il y aura tout autant d'attributs booléens correspondant. Par exemple, si l'attribut possède les valeurs A, B et C, alors trois nouveaux attributs booléens seront créés. Ces attributs seront de la forme « species » + « name », i.e. $\mathtt{a}_{speciesA}$, $\mathtt{a}_{speciesB}$ et $\mathtt{a}_{speciesC}$.

La figure 4.8b illustre la migration d'une instance $\mathtt{i}_{pest1}$ possédant 2 attributs multi-valués $\mathtt{a}_{species}$ et $\mathtt{a}_{family}$. Le premier, $\mathtt{a}_{species}$, représente l'espèce et possède la valeur « FuOx » (pour *Fusarium oxysporum*). Cet attribut est transformé en un ensemble d'attributs booléens, parmi lesquels on retrouve $\mathtt{a}_{speciesFuOx}$ et $\mathtt{a}_{SpLi}$ (pour *Spodoptera litura*) ainsi que toutes les autres valeurs possibles de l'attribut $\mathtt{a}_{species}$ (symbolisées par les « ... »). Pour

signifier que $\mathtt{i}_{pest1}$ possédait la valeur FuOx, on donne à l'attribut booléen correspondant (i.e. $\mathtt{a}_{speciesFuOx}$) la valeur `true` et à tous les autres la valeur `false`. On effectue les mêmes opérations pour l'attribut $\mathtt{a}_{family}$ représentant la famille de $\mathtt{i}_{pest1}$. Seul l'attribut correspondant à la valeur d'origine (i.e. $\mathtt{a}_{familyNectriaceae}$) se voit attribuer la valeur `true`.



(a) *Scaling d'attribut : Transformation du modèle. L'attribut multi-valué est transformé en un ensemble d'attributs booléens. Il y a autant d'attributs que de valeurs pour l'attribut multi-valué (e.g. A, B et C).*

(b) *Scaling d'attribut : Migration d'instance. Seul l'attribut correspondant à la valeur d'origine prend pour valeur* `true`*, les autres sont initialisés avec la valeur* `false`*.*

FIGURE 4.8 – *Scaling : transformation du modèle (à gauche) et migration des instances (à droite).*

Migration 5 : Le scaling d'attribut. *Soient* $\mathtt{M}_1$ *et* $\mathtt{M}_2$ *les diagrammes de classes avant et après la transformation. Soit* $\mathtt{C}^{\mathtt{M}_1}$ *une classe de* $\mathtt{M}_1$*.*

- *Chaque attribut* $\mathtt{a}^{\mathtt{M}_1}$ *de* $\mathtt{C}^{\mathtt{M}_1}$ *devient un ensemble d'attributs* $\mathtt{set}_{\mathtt{a}}^{\mathtt{M}_2}$ *dans* $\mathtt{C}^{\mathtt{M}_2}$*.*
- *Pour chaque* $\mathtt{set}_{\mathtt{a}}^{\mathtt{M}_2}$*, l'attribut correspondant à la valeur de* $\mathtt{a}^{\mathtt{M}_1}$ *prend la valeur* ***`true`****, les autres prennent la valeur* ***`false`****.*

4.1.3 Étape 3 : Conversion du diagramme de classes UML modifié vers une RCF

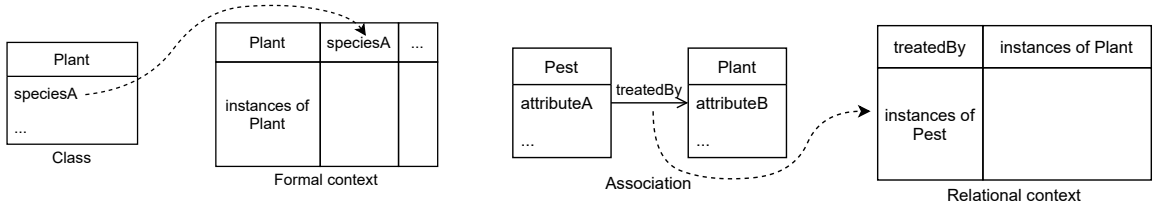
Une fois le modèle UML_{CoBUBo} obtenu et donc conforme aux restrictions d'une RCF, ses éléments sont convertis en éléments d'une RCF. La table 4.3 indique les conversions appliquées à chaque élément. Un contexte formel $\mathcal{K}$ est généré à partir de chaque classe `C`. Pour chaque classe `C`, chaque attribut booléen devient un attribut formel du contexte $\mathcal{K}$ correspondant. De même, un contexte formel r est généré à partir de chaque association `r`.

UML$_{\text{CoBUBo}}$ element to convert	**Formal context**	**Relational context**	**Description**
Class	x		→ 1 formal context $\mathcal{K}$ per class `C`, with $\mathcal{K}$'s name = `C`'s name
Attribute	x		→ attribute of the corresponding formal context
Association		x	→ 1 relational context r per association `r`, with r's name = `r`'s name

TABLE 4.3 – *Opérations à appliquer pour convertir les éléments du diagramme de classes UML en éléments d'une RCF.*

La figure 4.9 illustre ces conversions. La sous-figure 4.9a illustre la conversion de classe et d'attribut. La classe $\mathtt{C}_{Plant}$ devient le contexte formel $\mathcal{K}_{Plant}$ et ses attributs deviennent des attributs formels de $\mathcal{K}_{Plant}$. La sous-figure 4.9b illustre la conversion d'une association. L'association $\mathtt{r}_{treatedBy}$ devient le contexte relationnel $r_{treatedBy}$. Les instances de $\mathtt{C}_{Pest}$

(classe source de l'association) sont reportées dans la partie gauche du contexte $r_{treatedBy}$. Les instances de $\mathtt{C}_{Plant}$ (classe cible de l'association) sont reportées dans la partie haute du contexte $r_{treatedBy}$.



(a) *Transformation des classes et attributs.*

(b) *Transformation des associations.*

FIGURE 4.9 – *Transformation du diagramme UML_{CoBUBo} en une RCF. (a) Chaque classe devient un contexte formel et chaque attribut UML devient un attribut du contexte formel correspondant. (b) Chaque association UML devient un contexte relationnel.*

Après la conversion des éléments du modèle en contextes formels et relationnels, il faut ensuite convertir les instances et liens de chacun d'entre eux pour remplir les contextes. La figure 4.10 illustre un ensemble de six instances : trois instances de $\mathtt{C}_{Pest}$ et trois instances de $\mathtt{C}_{Plant}$. Chacune de ces instances est reliée à une ou plusieurs instances de l'autre classe via un ensemble de liens $\mathtt{l}_{Ri}$, i allant de 1 à 5. Par exemple, $\mathtt{i}_{to1}$ est en relation avec $\mathtt{i}_{pl2}$ par le lien $\mathtt{l}_{R4}$.

Chacune des trois instances des deux classes devient un objet $o \in G$ dans le contexte formel $\mathcal{K}$ correspondant. Ainsi $\mathtt{i}_{to1}$, $\mathtt{i}_{to2}$ et $\mathtt{i}_{to3}$ deviennent respectivement les première, deuxième et troisième lignes de $\mathcal{K}_{Pest}$ (sous-table 4.4a). De même pour $\mathtt{i}_{pl1}$, $\mathtt{i}_{pl2}$ et $\mathtt{i}_{pl3}$ qui deviennent respectivement les première, deuxième et troisième lignes de $\mathcal{K}_{Plant}$ (sous-table 4.4b). Les valeurs des attributs de chaque instance permettent quant à elles de remplir l'intension I du contexte formel. Chaque attribut ayant la valeur `true` est symbolisé par une croix dans la case correspondante du contexte qui représente la relation entre l'objet et son attribut.

Prototype logiciel implémentant la méthode Une première partie de prototype logiciel implémentant la méthode de conversion a été développée par Bachar Rima et

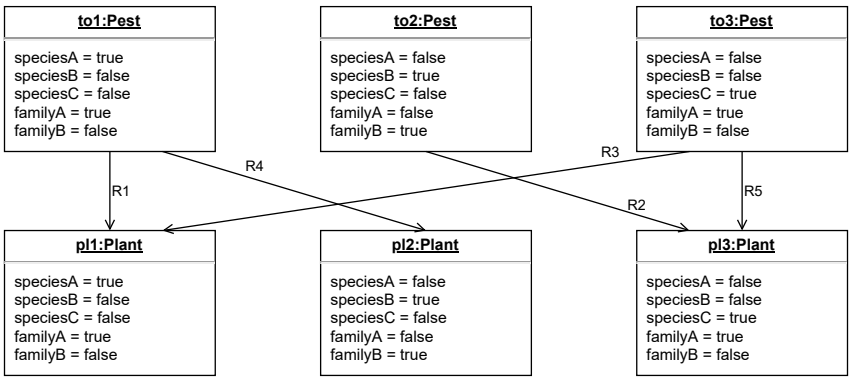


FIGURE 4.10 – *Exemple de diagramme d'objets UML dont les instances et liens doivent subir une conversion en éléments d'une RCF. Les instances de la classe $\mathtt{C}_{Pest}$ sont reliées aux instances de $\mathtt{C}_{Plant}$ grâce aux liens $\mathtt{l}_{Ri}$ ($i \in [1,5]$) de l'association $\mathtt{r}_{treatedBy}$.*

Pest	speciesA	speciesB	speciesC	familyA	familyB
to1	×			×	
to2		×			×
to3			×	×	

(a) *Contexte formel* $\mathcal{K}_{Pest}$

Plant	speciesAA	speciesBB	speciesCC	familyAA	familyBB
pl1	×			×	
pl2		×			×
pl3			×	×	

(b) *Contexte formel* $\mathcal{K}_{Plant}$

$r_{treatedBy}$	pl1	pl2	pl3
to1	×	×	
to2			×
to3	×		×

(c) *Contexte relationnel* $r_{treatedBy}$ *entre* $\mathcal{K}_{Pest}$ *et* $\mathcal{K}_{Plant}$

TABLE 4.4 – *Exemple de RCF correspondant au diagramme de classes de la figure 4.10. Chaque instance est devenue une ligne du contexte formel correspondant et chaque lien est devenu une* × *entre deux objets du contexte relationnel.*

Joseph Saba à l'occasion de leur projet de TER lors du Master Informatique. Il est réalisé avec des outils de l'ingénierie dirigée par les modèles (Eclipse Modeling Framework). Ce prototype met en œuvre notre méthode par le codage d'une transformation de modèle allant d'un modèle conforme au méta-modèle UML vers un modèle conforme au méta-modèle de RCF. Ce prototype est accessible sur un dépôt GitHub [1].

4.2 Diverses modélisations de la relation ternaire

L'élément central du modèle de données de Knomana est la relation ternaire (association ternaire dans le vocabulaire UML). Cette relation connecte trois composants principaux : le biopesticide, l'agresseur et l'organisme protégé. RCA ne permettant pas de représenter les relations ternaires, elle doit être convertie en un ensemble de relations binaires.

Pour cela, la solution la plus répandue, et recommandée par le W3C [Noy and Rector, 2006], consiste à matérialiser la relation ternaire en une classe reliée aux classes préexistantes. La figure 4.11 illustre cette transformation avec la matérialisation de la relation ternaire `protectionSystem` via la transformation Na_{mat} définie en section 4.1.1.

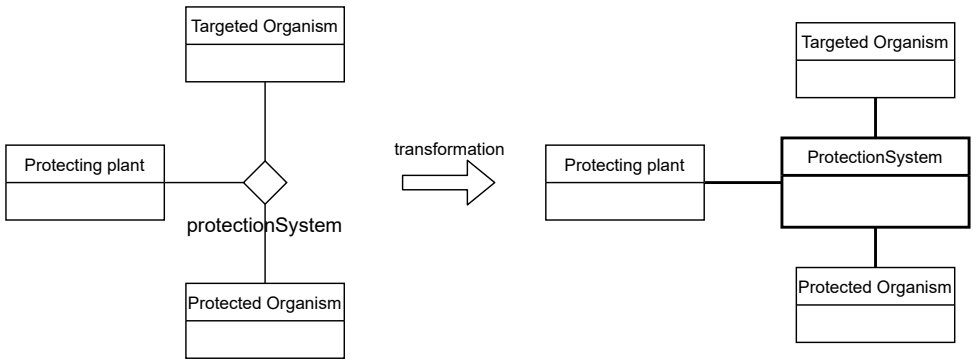


FIGURE 4.11 – *Méthode classique de transformation d'une relation n-aire. Cette relation est matérialisée dans une nouvelle classe, qui est ensuite reliée aux classes extrémités de la relation grâce à* n *nouvelles associations.*

1. https://github.com/anonbnr/UML2RCA

Dans le cadre du projet, il peut être intéressant pour l'utilisateur de modifier la relation ternaire de façon à disposer de structures de présentation des connaissances qui soient plus à même de répondre à ses questions. Ce faisant, il peut souhaiter appliquer des transformations produisant des modèles qui ne soient pas strictement équivalents au modèle initial. La figure 4.12 illustre plusieurs modélisations de la relation ternaire utilisées au cours de la thèse. On y retrouve tout d'abord la matérialisation de la relation ternaire, i.e. la transformation Na_{mat} (en haut à gauche). La deuxième (en bas à gauche) est une représentation qui relie chaque type d'organisme aux autres, formant ainsi un graphe complet (il s'agit de la transformation Na_{split}). Ainsi, le biopesticide fabriqué à partir d'une plante (P_plant) est en relation avec l'agresseur (Pest) et avec l'organisme protégé (Crop). L'agresseur est en relation avec la plante et l'organisme protégé. Et enfin l'organisme protégé est en relation avec la plante et l'agresseur. Avec ce modèle, il n'est pas indiqué que le produit fabriqué à partir d'une plante est utilisé pour protéger un organisme contre un agresseur. Seules les relations élémentaires sont connues. La troisième (en bas à droite) est une représentation en chaîne où l'organisme protégé est protégé par une plante qui traite un agresseur. En fonction de l'élément qui intéresse le plus l'utilisateur pour répondre à une question, l'ordre des éléments de la chaîne peut être modifié, impliquant *de facto* des classifications différentes. La quatrième (en haut à droite) est un ensemble de relations binaires entre deux organismes (crop et pest) dont chacune est étiquetée avec un objet du troisième organisme (plant). Par exemple, nous avons choisi d'étiqueter la relation entre les cultures protégées et les agresseurs par chacune des plantes servant à produire un biopesticide. En fonction de l'élément que l'utilisateur souhaite faire ressortir, il peut adapter la configuration de la conversion.

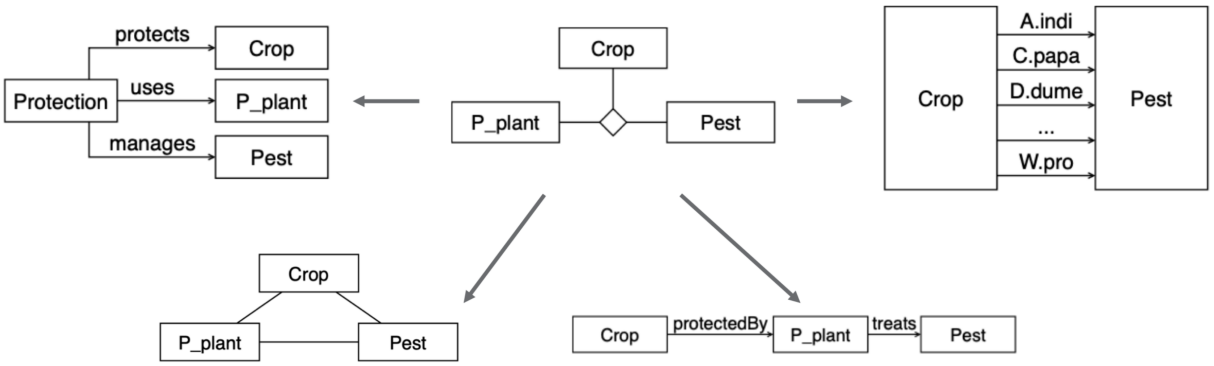


FIGURE 4.12 – *Différentes modélisations de la relation n-aire : (en haut à gauche) la conversion classique qui matérialise la relation n-aire, (en haut à droite) un ensemble de relations binaires entre deux des entités. Ces relations sont étiquetées par les objets de la troisième. En bas à gauche, un graphe complet entre les n entités et en bas à droite une chaîne. Pour les deux représentations de droite, la place de chacune des entités peut être modifiée.*

4.3 Modélisation des données indéterminées

La base de connaissances Knomana rassemble des connaissances issues de la littérature scientifique. Parmi les données recensées, il y en a certaines pour lesquelles la valeur de la donnée n'est pas précise ; ce qui lui confère le statut de donnée à valeur indéterminée. Plusieurs exemples ont été présentés en section 1.1.1. Par exemple, l'utilisation du nom

vernaculaire « piment » fait référence aux fruits produits par cinq espèces de plante du genre *Capsicum*. Dans la mesure où ce genre en compte quarante-deux d'après le site web Catalogue Of Life, utiliser le terme Piment n'est pas suffisant pour connaître précisément le nom scientifique de l'espèce de plante d'où provient le fruit indiqué. De la même façon, la dénomination scientifique peut comporter des valeurs indéterminées lors de l'indication du nom de l'espèce, en utilisant notamment les abréviations « sp. » et « spp. ». Sp. indique que le nom donné correspond à une seule espèce du genre, dont on ne connaît pas le nom (e.g. *Spodoptera sp.*). Spp. fait référence à au moins deux espèces du genre (e.g. *Spodoptera spp.*), i.e. dont on ne connaît ni le nom des espèces ni leur nombre. Cette section s'intéresse à la modélisation du cas d'indétermination « spp. ».

Nous avons vu dans l'état de l'art (section 3.4.2) que la taxinomie permet de compléter la liste des attributs d'un contexte formel via la relation d'héritage. Dans le cas de « spp. », la relation n'est pas une relation de spécialisation/généralisation car « spp. » représente au moins deux espèces d'un genre qui ne sont pas mentionnées explicitement. De façon à modéliser cette situation, nous proposons d'opérer un scaling qui lève l'indétermination en regard du contenu de la base de connaissances. Il consiste à attribuer à une espèce nommée au moyen de spp. les attributs associés aux autres organismes du même genre selon une agrégation graduelle, que l'on appelle hiérarchie dans cette section, et réciproquement. L'objet de cette hiérarchie est de permettre à l'utilisateur de comprendre la classification des connaissances présentées au cours de l'exploration, et d'identifier les potentielles erreurs de regroupement ou de noms d'espèces probables.

La figure 4.13 présente un exemple de hiérarchie concernant 3 espèces du genre *Spodoptera* (S. est la contraction de *Spodoptera*). Deux espèces sont connues, i.e. *S. litu* et *S. litto*, et la troisième est indéterminée *S. spp.* Au bas de la hiérarchie, se trouvent les données de base. A ce niveau S. spp. est donc considéré comme une espèce au même titre que S. litto ou S. litu. Au premier niveau de la hiérarchie, deux nouvelles espèces sont créées combinant les données relatives à une espèce et relatives à spp. Enfin, une nouvelle espèce est créée au second niveau, qui combine toutes les données des trois espèces (appelée S.spp+any). Cette fois-ci, les données de chaque espèce sont dupliquées dans S. spp. On peut remarquer que le premier niveau d'agrégation peut également être adopté pour l'abréviation sp. Au lieu de considérer au moins deux espèces incluant celle spécifiée, il s'agit d'une espèce étant potentiellement celle spécifiée. Avec ce type de scaling appliqué à l'abréviation spp., il y a un total de $2 \times n$ espèces associées à ce genre, n étant le nombre original d'espèces.

4.4 Exemples caractéristiques des transformations de modèles UML en RCF

Cette section illustre certaines transformations présentées précédemment sur de petits exemples. Pour chaque illustration, les instances de diagramme de classes UML comportent des attributs booléens. Les contextes formels, produits à partir de ces instances, mettent en évidence la correspondance entre le modèle UML et sa conversion en RCF.

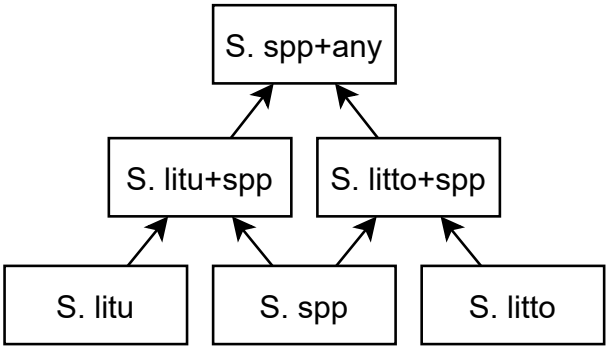


FIGURE 4.13 – *Exemple d'application du scaling permettant de lever l'indétermination du nom d'espèce utilisant l'abréviation spp. Le niveau bas comporte les données de base. Le niveau moyen regroupe les données de chaque espèce avec celle de spp. Le niveau haut regroupe les données de toutes les espèces.*

Conversion d'une association birectionnelle en deux associations unidirectionnelles ($\mathbf{Bi_{2uni}}$). La figure 4.14 présente des instances des classes $\mathtt{C}_{Pest}$ et $\mathtt{C}_{Plant}$, de même que les liens R1 à R5 entre ces instances. Les contextes formels sont remplis à l'aide de ces instances. Le tableau 4.5 présente les contextes formels renseignés à partir de ces instances. Par exemple, l'instance $\mathtt{i}_{to1:Pest}$ est convertie pour donner la première ligne de la sous-table 4.5a.

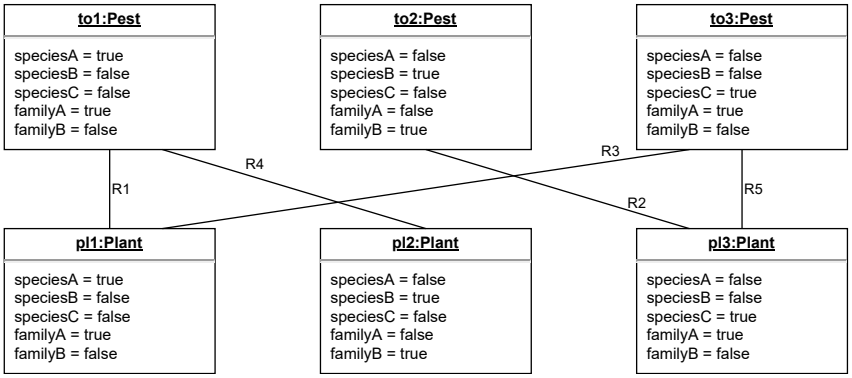


FIGURE 4.14 – *Exemple de diagramme d'objets UML pour $\mathtt{C}_{Pest}$ et $\mathtt{C}_{Plant}$ reliées par une association bidirectionnelle.*

Pest	speciesA	speciesB	speciesC	familyA	familyB
to1	×			×	
to2		×			×
to3			×	×	

(a) *Contexte formel $\mathcal{K}_{Pest}$*

Plant	speciesAA	speciesBB	speciesCC	familyAA	familyBB
pl1	×			×	
pl2		×			×
pl3			×	×	

(b) *Contexte formel $\mathcal{K}_{Plant}$*

TABLE 4.5 – *Contextes formels $\mathcal{K}_{Pest}$ (a) et $\mathcal{K}_{Plant}$ (b) résultant de la transformation correspondant au diagramme de la figure 4.14 (en appliquant un scaling nominal).*

Les objets de $\mathtt{C}_{Pest}$ et $\mathtt{C}_{Plant}$ sont en relation tel que le décrit la Figure 4.14. Chaque relation est bidirectionnelle. En appliquant $\mathrm{Bi}_{2\mathrm{uni}}$, deux associations unidirectionnelles sont créées, et donc deux contextes relationnels sont produits (cf. table 4.6). La sous-table 4.6a présente le contexte relationnel $r_{treatedBy}$ allant des objets de $\mathcal{K}_{Pest}$ vers les

objets de $\mathcal{K}_{Plant}$, et la sous-table 4.6b présente son inverse, i.e. le contexte relationnel allant des objets de $\mathcal{K}_{Plant}$ vers les objets de $\mathcal{K}_{Pest}$. Ainsi, le lien $\mathtt{l}_{R1}$ entre les instances $\mathtt{i}_{to1}$ et $\mathtt{i}_{pl1}$ est représenté dans chacun des deux contextes relationnels $r_{treatedBy}$ et r_{treats} par la croix dans la case située à la jonction des lignes des deux objets. Dans la suite de cette section, seul le premier sens de la relation est présenté. Le contexte relationnel obtenu dans le second sens est appelé « contexte inverse ».

$r_{treatedBy}$	**pl1**	**pl2**	**pl3**
to1	×	×	
to2			×
to3	×		×

(a) *Contexte relationnel $r_{treatedBy}$ entre $\mathcal{K}_{Pest}$ et $\mathcal{K}_{Plant}$*

r_{treats}	**to1**	**to2**	**to3**
pl1	×		×
pl2	×		
pl3		×	×

(b) *Contexte relationnel r_{treats} entre $\mathcal{K}_{Plant}$ et $\mathcal{K}_{Pest}$*

TABLE 4.6 – *Contextes relationnels $r_{treatedBy}$ (a) et r_{treats} (b) correspondant à l'association bidirectionnelle entre $\mathcal{K}_{Pest}$ et $\mathcal{K}_{Plant}$ illustrée par la figure 4.14.*

Conversion d'une classe d'association ($\mathbf{Att_{mat}}$) Dans la figure 4.14, nous considérons que chaque lien a un attribut le caractérisant. En appliquant $\mathrm{Att}_{\mathrm{mat}}$, une nouvelle classe est ajoutée et elle a autant d'instances qu'il y a de liens, et deux associations sont créées entre les extrémités de la relation et la nouvelle classe. La table 4.7 présente les contextes correspondants produits. La sous-table 4.7a est le contexte formel représentant la nouvelle classe. Chaque lien de l'association est converti en une ligne du contexte formel. Les contextes relationnels 4.7b et 4.7c et leurs contextes inverses sont les contextes produits représentant les associations. Chaque objet de $\mathcal{K}_{Pest}$ ou $\mathcal{K}_{Plant}$ est mis en relation dans les contextes relationnels avec l'objet de $\mathcal{K}_{TreatedBy}$, correspondant à un lien de l'association qui le mentionne.

TreatedBy	**toxic**	**repulsive**
R1	×	
R2	×	
R3	×	
R4		×
R5	×	

(a) *Contexte formel $\mathcal{K}_{TreatedBy}$ représentant une classe d'association*

$r_{Pest-TreatedBy}$	**R1**	**R2**	**R3**	**R4**	**R5**
to1	×			×	
to2		×			
to3			×		×

(b) *Contexte relationnel entre $\mathcal{K}_{Pest}$ et $\mathcal{K}_{TreatedBy}$*

$r_{Plant-TreatedBy}$	**R1**	**R2**	**R3**	**R4**	**R5**
pl1	×		×		
pl2				×	
pl3		×			×

(c) *Contexte relationnel entre $\mathcal{K}_{Plant}$ et $\mathcal{K}_{TreatedBy}$*

TABLE 4.7 – *Conversion finale d'une classe d'association UML. La classe d'association est matérialisée par $\mathcal{K}_{TreatedBy}$ (a) et est reliée aux extrémités de la l'association ($\mathcal{K}_{Pest}$ et $\mathcal{K}_{Plant}$) par les contextes relationnels (b) et (c) ainsi que leur contexte inverse.*

4.4.1 Exemple de conversion d'une association n-aire

La section 4.1 a défini et formalisé deux manières de transformer une association n-aire au moyen des transformations $\mathrm{Na}_{\mathrm{mat}}$ et $\mathrm{Na}_{\mathrm{split}}$. La table 4.8 montre les instances de trois classes (déjà converties en contextes formels) qui sont en relation grâce à une

association n-aire illustrée par la figure 4.15. Les objets de la sous-table 4.8a représentent les organismes ciblés par les objets de 4.8b pour protéger les objets de 4.8c.

Targeted	speciesQ	speciesS	speciesD	familyQ	familyS
to1	×			×	
to2		×			×
to3			×	×	

(a) *Contexte formel* $\mathcal{K}_{Targeted}$

Plant	speciesA	speciesZ	speciesE	familyA	familyZ
pl1	×			×	
pl2		×			×
pl3			×	×	

(b) *Contexte formel* $\mathcal{K}_{Plant}$

Protected	speciesW	speciesX	speciesC	familyW	familyX
po1	×			×	
po2		×			×
po3			×	×	

(c) *Contexte formel* $\mathcal{K}_{Protected}$

TABLE 4.8 – *Exemple de trois contextes formels correspondant à trois classes* $\mathtt{C}_{Plant}$, $\mathtt{C}_{TargetedOrganism}$, *and* $\mathtt{C}_{ProtectedOrganism}$.

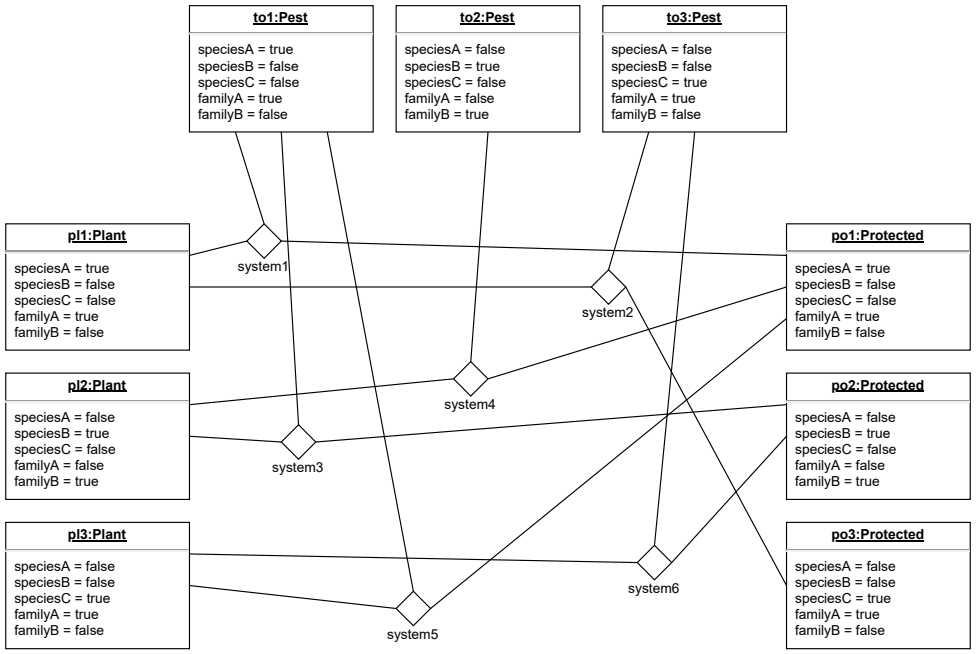


FIGURE 4.15 – *Diagramme d'objets présentant les liens entre les instances des classes* $\mathtt{C}_{Plant}$, $\mathtt{C}_{Targeted}$ *and* $\mathtt{C}_{Protected}$ *à l'origine de la table 4.8.*

L'application de $\mathrm{Na}_{\mathrm{mat}}$ produit les contextes de la table 4.9. L'association matérialisée est représentée par $\mathcal{K}_{Protection}$ (sous-table 4.9a). Les n associations sont représentées par les contextes relationnels des sous-tables 4.9b, 4.9c et 4.9d et leurs contextes inverses. De la même manière que pour la classe d'association, chaque objet de $\mathcal{K}_{Targeted}$, $\mathcal{K}_{Plant}$ ou $\mathcal{K}_{Protected}$ est mis en relation avec l'objet de $\mathcal{K}_{Protection}$ correspondant à un lien de l'association le mentionnant.

L'application de $\mathrm{Na}_{\mathrm{split}}$ produit un graphe complet entre les n classes de l'association. Dans cet exemple, il produit les trois contextes relationnels présentés dans la table 4.10 et leurs contextes inverses. Chaque objet d'un contexte formel est connecté à un objet de chacun des deux autres contextes formels (cette connexion est effectuée par des paires d'objets ; les objets sont donc connectés deux à deux et non par un triplet), entraînant une perte d'information sur le lien entre les trois objets. En utilisant ces contextes relationnels produits, nous savons que `pl1` traite `to1` et `to3` et protège `po1` et `po3`. Mais il n'est pas dit si `pl1` traite `to1` pour protéger `po1` ou `po3`. Il y a donc une perte d'information. Cette perte ouvre cependant la possibilité de formuler de nouvelles hypothèses.

$\mathcal{K}_{ProtectionSystem}$
system1
system2
system3
system4
system5
system6

(a) *Contexte formel* $\mathcal{K}_{Protection}$

r_{uses}	**pl1**	**pl2**	**pl3**
system1	×		
system2	×		
system3		×	
system4		×	
system5			×
system6			×

(b) *Contexte relationnel* r_{uses}

$r_{targets}$	**to1**	**to2**	**to3**
system1	×		
system2			×
system3	×		
system4		×	
system5	×		
system6			×

(c) *Contexte relationnel* $r_{targets}$

$r_{protects}$	**po1**	**po2**	**po3**
system1	×		
system2			×
system3		×	
system4	×		
system5	×		
system6		×	

(d) *Contexte relationnel* $r_{protects}$

TABLE 4.9 – *Conversion finale d'une association n-aire avec la transformation* Na_{mat}. *Les instances et liens utilisés sont respectivement présentés dans les tables 4.8 et la figure 4.15. L'association est matérialisée par* $\mathcal{K}_{ProtectionSystem}$ *et est reliée aux extrémités de l'association (*$\mathcal{K}_{Pest}$, $\mathcal{K}_{Plant}$ *et* $\mathcal{K}_{Protected}$*) par les contextes relationnels (b), (c) et (d) ainsi que leur contexte inverse.*

r_{treats}	**to1**	**to2**	**to3**
pl1	×		×
pl2	×	×	
pl3	×		×

(a) *Contexte relationnel* r_{treats}

$r_{protects}$	**po1**	**po2**	**po3**
pl1	×		×
pl2	×	×	
pl3	×	×	

(b) *Contexte relationnel* $r_{protects}$

$r_{attacks}$	**po1**	**pot2**	**po3**
to1	×	×	
to2	×		
to3		×	×

(c) *Contexte relationnel* $r_{attacks}$

TABLE 4.10 – *Conversion finale d'une association n-aire avec la transformation* Na_{split}. *Chacun des trois contextes relationnels et leurs contextes inverses représentent les liens entre les objets conformément à la figure 4.15.*

4.4.2 Exemple de conversion de l'héritage

Pour convertir la relation d'héritage d'un diagramme de classes UML dans une RCF, il faut d'abord choisir un niveau de représentation. Considérons le diagramme de la figure 4.16 et choisissons la classe $\mathtt{C}_{Article}$ comme niveau de représentation. Les instances de $\mathtt{C}_{Plant}$ sont toujours celles situées en bas de la figure 4.10. Les instances correspondant à la hiérarchie du diagramme de la figure 4.16 sont : les instances $\mathtt{i}_{art}$ et $\mathtt{i}_{jart}$ de la figure 4.3 ainsi que trois documents hypothétiques (`doc1`, `doc2` et `doc3`). En choisissant $\mathtt{C}_{Article}$ comme niveau de représentation et en appliquant les transformations concernant l'héritage, on obtient les contextes formels et relationnels de la table 4.11. L'article étant une sous-classe de $\mathtt{C}_{Document}$, les articles sont des documents. Ainsi, les objets `art` et `jart` sont dupliqués dans les deux contextes formels. Cela permet également de représenter la généralisation dans la RCF car les objets dupliqués ont ainsi plusieurs types, l'un d'eux étant plus spécifique qu'un autre.

4.4.3 Exemple de conversion pour des données indéterminées

Dans notre méthode, la conversion des données indéterminées passe par l'utilisation d'un scaling particulier illustré par la figure 4.13 de la section 4.3. Considérons trois instances de $\mathtt{C}_{Pest}$ telles que « S. litto », « S. litu » et « S. Spp » dont les seuls attributs sont leurs identifiants eux-mêmes. Avec le scaling défini, on peut ainsi obtenir l'un ou l'autre des contextes formels de la table 4.12 en fonction de l'application du scaling dans

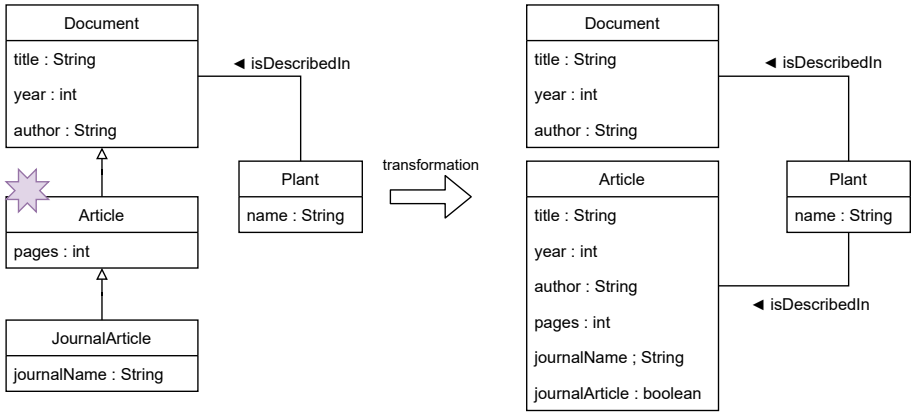


FIGURE 4.16 – *Exemple de transformation d'un modèle UML contenant de l'héritage (copie de la figure 4.2)*

Document	titleA	titleB	titleC	titleD	titleE	authorA	authorB	yearA	yearB
doc1	×					×		×	
doc2		×				×			×
doc3			×				×	×	
art1				×		×			×
jart1					×		×		×

(a) *Contexte formel* $\mathcal{K}_{Documents}$

$r_{isDescribedIn}$	doc1	doc2	doc3	art	jart
pl1	×			×	×
pl2			×		×
pl3		×		×	×

(b) *Contexte relationnel* $r_{isDescribedIn}$ *entre* $\mathcal{K}_{Plant}$ *et* $\mathcal{K}_{Document}$

Article	titleA	titleB	titleC	titleD	titleE	authorA	authorB	yearA	yearB	pagesA	pagesB	journalNameA	JournalArticle
art2				×		×			×	×			
jart2					×		×		×		×	×	×

(c) *Contexte formel* $\mathcal{K}_{Article}$

$r_{isDescribedIn}$	art	jart
pl1	×	×
pl2		×
pl3	×	×

(d) *Contexte relationnel* $r_{isDescribedIn}$ *entre* $\mathcal{K}_{Plant}$ *et* $\mathcal{K}_{Article}$

TABLE 4.11 – *Conversion en RCF de la figure 4.16. L'instance* $\mathtt{i}_{art}$ *est dupliquée dans* $\mathcal{K}_{Document}$ *(art1) et* $\mathcal{K}_{Article}$ *(art2). De même pour* $\mathtt{i}_{jart}$ *avec jart1 et jart2. En combinant les informations présentes dans tous les contextes, on est donc capable de retrouver la notion d'héritage.*

les attributs ou dans les instances de $\mathtt{C}_{Pest}$ (et donc dans ceux de $\mathcal{K}_{Pest}$).
La sous-table 4.12a illustre l'application du scaling dans les attributs où chacun des objets possède, en plus de ses propres attributs, les nouveaux attributs qui lui correspondent. Par exemple, l'objet « S. litto » possède son attribut identifiant « S. litto » et les deux attributs supplémentaires « S. litto+spp » et « S. spp+any ».
La sous-table 4.12b illustre l'application du scaling dans les objets où sont ajoutés des nouveaux objets pour représenter « S. litto+spp », « S. litu+spp » et « S. spp+any ». Chacun de ces trois objets possédera les attributs des objets qu'il regroupe. Par exemple, l'objet « S. litto+spp » possède les attributs de « S. litto » et de « S. Spp ».

$\mathcal{K}_{Pest}$	**S. litto**	**S. litu**	**S. Spp**	**S. litto+spp**	**S. litu+spp**	**S. spp+any**
S. litto	×			×		×
S. litu		×			×	×
S. Spp			×	×	×	×

(a) *Contexte formel $\mathcal{K}_{Pest}$ suite à un scaling sur les attributs pour les données indéterminées*

$\mathcal{K}_{Pest}$	**S. litto**	**S. litu**	**S. Spp**
S. litto	×		
S. litu		×	
S. Spp			×
S. litto+spp	×		×
S. litu+spp		×	×
S. spp+any	×	×	×

(b) *Contexte formel $\mathcal{K}_{Pest}$ suite à un scaling sur les objets pour les données indéterminées*

TABLE 4.12 – *Conversion en RCF du scaling pour les données indéterminées appliqué dans les attributs (à gauche) ou dans les objets (à droite)*

4.5 Conclusion

Ce chapitre a proposé, dans un cadre unifié, une méthode de conversion d'un jeu de données dont le modèle de classes est au format UML en une famille de contextes relationnels (RCF). Cette méthode, complétée d'un algorithme, convertit des éléments tels que les classes, les attributs, les associations binaires simples et les instances qui sont présents dans la majorité des diagrammes de classes UML. Chacune des opérations est définie en détail avec une application de l'opération sur un exemple, suivie d'une formalisation mathématique (section 4.1).

En outre, notre méthode propose des solutions de modélisation pour les associations n-aires et les données indéterminées. Pour finir, en nous inspirant des approches du génie logiciel et de la science des données, notre méthode permet la conversion de l'héritage pour les langages dans lesquels cette notion n'est pas native. Nous introduisons aussi un algorithme qui met en œuvre les différentes conversions.

Par la suite, ce chapitre a présenté des solutions de modélisation de la relation ternaire (section 4.2), puis un scaling permettant de lever l'indétermination des données en lien avec le nom des espèces (section 4.3). Ce chapitre présente enfin, en section 4.4, des exemples permettant une meilleure compréhension des différentes conversions. Il n'y a pas d'équivalent, dans la littérature, d'un tel algorithme ni d'un tel regroupement de toutes ces opérations, dans un cadre de formalisation unifié.

Le chapitre suivant présente les évaluations quantitatives et qualitatives de ces conversions sur des cas d'étude abordés au cours de la thèse. De plus, afin de montrer la pertinence de ces conversions, des évaluations sur les structures conceptuelles produites à partir des résultats des conversions sont également présentées.

V

Évaluation des conversions

Les modifications produites par les opérations de conversion présentées au chapitre précédent ont un impact sur les données d'entrée de la méthode de classification choisie et sur ses résultats. Au cours de la thèse, nous avons eu l'occasion de faire diverses expérimentations sur ces conversions avec l'Analyse de Concepts Formels en prenant pour cas d'étude des extraits de la base de connaissances Knomana.

Ce chapitre présente ces expérimentations. La section 5.1 présente les différents critères et indicateurs qui ont guidé les évaluations. Des évaluations quantitatives sur les RCF obtenues grâce aux conversions sont présentées dans la section 5.2. Des évaluations quantitatives et qualitatives permettent d'examiner la pertinence des structures conceptuelles produites à partir du résultat des conversions (section 5.3). Nous analysons la validité expérimentale en section 5.4. Puis nous concluons en section 5.5.

Logiciel et matériel utilisés pour obtenir les résultats des cas d'utilisation Pour appliquer RCA sur nos données, nous avons tout d'abord utilisé le logiciel RCAexplore [1] qui a été développé pour explorer les données hydroécologiques relationnelles lors du projet ANR 11 MONU 14 Fresqueau. À chaque itération du processus RCA, RCAeplore permet de choisir : l'algorithme pour construire les structures conceptuelles (AOC-poset, etc.), le quantifieur (comme le quantifieur existentiel $\exists$) et de sélectionner des contextes formels et relationnels. Nous avons ensuite utilisé les algorithmes de FCA et de RCA intégrés par Alain Gutierrez dans la plateforme CoGui [2]. Cette plateforme est développée dans l'équipe GraphiK (LIRMM) et dédiée à la représentation et au raisonnement à base de graphes. Des algorithmes d'extraction et de conversion spécialisés ont de plus été implémentés en Java. Puis Alain Gutierrez a créé la bibliothèque FCA4J [3] en y intégrant la plupart des algorithmes. Ce sont ces outils qui peuvent permettre de répliquer les expérimentations. Celles-ci ont été réalisées à l'aide d'un ordinateur portable équipé d'un processeur 4 cœurs

1. http://dataqual.engees.unistra.fr/logiciels/rcaExplore
2. http://www.lirmm.fr/cogui/
3. https://www.lirmm.fr/fca4j/

Intel i7 2,70 GHz.

Nota Dans ce chapitre nous utilisons indifféremment les termes de conversion ou de transformation.

5.1 Définition des critères d'évaluation

Afin d'évaluer notre approche, trois critères ont été identifiés : (i) l'applicabilité des transformations proposées en section 4.1, (ii) le passage à l'échelle, i.e. la capacité des résultats à être exploitables par les méthodes de classification suite à l'augmentation de volume entraînée par les conversions, et (iii) la pertinence des données converties. Pour mesurer ces critères, des indicateurs (ou métriques) ont été définies. Cette section présente les critères puis les métriques.

L'applicabilité *Les transformations sont-elles applicables à des données existantes ?* Notre hypothèse est que, dans le domaine de la santé et de l'environnement, beaucoup de jeux de données contiennent une complexité intrinsèque qui demande à être convertie pour être explorée. Que ce soit pour appliquer FCA ou toute autre méthode demandant des transformations de données similaires. Cette complexité peut apparaître sur différents aspects : au niveau des données (avec des données numériques ayant différentes échelles ou décrites dans des taxonomies) ou au niveau du modèle de données (avec des relations n-aires ou des hiérarchies de spécialisation/généralisation). Au travers de nos cas d'étude, qui sont des situations réelles, nous évaluerons s'il existe des situations nécessitant ce framework de conversion.

Le passage à l'échelle (*scalability* en anglais) *Les données converties sont-elles exploitables par les méthodes basées sur FCA ?* La conversion modifie la taille des données à analyser. Par exemple, on peut obtenir un jeu beaucoup plus volumineux qu'à l'origine en convertissant des attributs multi-valués en attributs booléens. Les méthodes de classification basées sur FCA sont connues pour conduire potentiellement à une explosion combinatoire. Ainsi, une augmentation des données d'entrée peut avoir un fort impact sur le résultat. Pour évaluer ce risque, nous observons, dans un premier temps, la taille de l'ensemble de données après application des transformations. Puis, nous observons la taille de la sortie de ces méthodes basées sur FCA et le temps de calcul de ces méthodes. En effet, cette sortie doit être exploitable par des experts ; c'est-à-dire qu'ils doivent pouvoir en extraire des connaissances à l'aide de méthodes ou d'outils de visualisation et de navigation dans les structures conceptuelles (treillis, Iceberg, AOC-poset).

La pertinence *Les données converties sont-elles pertinentes pour les méthodes basées sur FCA ?* Nous voulons savoir si les données, converties pour FCA et ses extensions, contiennent toujours des informations utiles aux experts. Par exemple, lorsque des attributs multi-valués sont transformés en attributs booléens, puis classés au sein de structures

conceptuelles, les experts acquièrent-ils encore des connaissances pertinentes ? Nous répondrons à cette question par une analyse qualitative.

Indicateurs (métriques) La mesure de l'applicabilité est effectuée en utilisant différents indicateurs pour les associations n-aires et l'héritage. Les deux indicateurs pour les associations n-aires sont le nombre d'associations (*#n-ary*) dans les modèles de données et le nombre de triplets impliqués (*#tri*). Les quatre indicateurs de généralisation sont le nombre d'instances impliquées dans la hiérarchie (*#doc*), la profondeur de la hiérarchie représentée (*#lvl*), le nombre de relations d'héritage (*#ge-sp*) et la largeur de la hiérarchie (*#nc-cl*), i.e. le nombre maximum de classes non comparables.
La mesure du passage à l'échelle se fait à l'aide de deux indicateurs sur les RCFs produites : le nombre d'objets par contexte formel (*#objects*) et le nombre d'attributs booléens total par contexte formel (*#attributes*).

5.2 Évaluation des conversions

Il peut être intéressant, pour les experts, de se focaliser sur un extrait des connaissances pour répondre à un besoin particulier. Cela implique de préciser le modèle de données afférent. Les divers cas d'étude abordés au cours de la thèse ont été sélectionnés en fonction de considérations variées. L'un d'entre eux se focalise sur les références bibliographiques ayant servi au recensement des connaissances. D'autres ont été constitués autour de plantes de protection sélectionnées par des experts pour leur efficacité dans diverses applications et leur présence en Afrique. Si chacun de ces cas se focalise sur un petit aspect de Knomana, il est également intéressant de voir ce qui se passe quand on veut convertir un extrait beaucoup plus conséquent de la base.

Chaque cas d'étude considère un jeu de données différent. Chacun de ces jeux est accompagné d'un modèle de données transformé en une RCF. Les jeux de données ayant été sélectionnés en fonction de certaines situations ou de certaines questions, lors de la conversion des modèles, certaines associations n'ont pas été conservées. La direction des associations conservées a été déterminée dans le but d'obtenir des classifications et une propagation des relations avec RCA en accord avec chacune des situations considérées. Par ailleurs, il est intéressant pour un expert de pouvoir choisir lui-même, lors du processus, s'il garde les deux associations unidirectionnelles produites ou seulement l'une d'elles. C'est notamment le cas pour les transformations des associations n-aires qui ne sont pas orientées. Cela permet également de réduire la présence de circuits dans le modèle final et donc d'avoir des structures plus petites. Enfin, les modèles présentés dans cette section sont les modèles UML à l'étape UML_{CoBUMu}, i.e. les modèles avant *scaling* des attributs multi-valués, afin de permettre un gain de place non négligeable sur les figures.

Dans cette section, les évaluations sont présentées comme suit. Premièrement, le jeu de données est décrit en termes de contenu et de dimension. Ensuite, le modèle de données correspondant est présenté. À partir de cela, les résultats de l'applicabilité sont donnés. Deuxièmement, les résultats des conversions sont présentés et expliqués. Ce qui permet enfin de donner les résultats du passage à l'échelle. Cette section ne contient pas de

résultat quantitatif sur la conversion des données indéterminées car le jeu utilisé avec cette conversion est trop petit et trop simple et donc, n'offre pas d'intérêt. Cet aspect est traité dans le cadre d'une évaluation qualitative dans laquelle diverses extensions de FCA sont comparées (section 5.3.3).

5.2.1 Évaluation quantitative pour l'héritage

La première partie de l'algorithme de conversion concerne l'héritage. Ce premier cas d'étude est donc constitué d'un jeu et d'un modèle de données permettant d'appliquer les transformations liées à l'héritage dans un diagramme de classes UML.

Cas d'étude sur les références bibliographiques de Knomana (DataDoc) Ce jeu de données concerne un ensemble de documents utilisés lors du recensement des connaissances dans la base Knomana. Il rassemble les rapports, thèses et articles (de journal ou de conférence) enregistrés dans Knomana concernant le jeu PPAf. Au total, il comprend 788 documents écrits par 642 auteurs entre 1958 et 2019 décrits par 7 attributs multi-valués : 3 communs à tous les documents (titre, année et premier auteur), 1 supplémentaire pour les thèses (le nom de l'école) et 3 communs à tous les articles (le nombre de pages et le nom du livre ou de la conférence).

Le modèle illustré par la Figure 5.1 est focalisé sur l'héritage. La classe abstraite $\mathtt{C}_{Document}$ regroupe 3 attributs communs à tous les documents, i.e. un titre, une année et un auteur (le premier auteur). Deux types de documents sont ensuite différenciés : les rapports ($\mathtt{C}_{Report}$) qui regroupent également les thèses ($\mathtt{C}_{Thesis}$, avec le nom de l'école) et les articles ($\mathtt{C}_{Article}$). Les articles, qui possèdent un nombre de pages déterminé, sont ensuite eux-mêmes différenciés en article de conférence ($\mathtt{C}_{InProceedings}$) ou en article de journaux ($\mathtt{C}_{JournalArticle}$). Chacun des deux types d'articles introduit respectivement le titre du livre et le nom du journal dans lequel il est publié.

Résultat de l'applicabilité DataDoc contient les 788 documents utilisés pour remplir PPAf. Ces documents sont organisés hiérarchiquement. La profondeur de cette hiérarchie est de 2. Elle contient 5 relations d'héritage et sa largeur est égale à 3. Ceci est formalisé ci-après :

$$\#doc = 788 \qquad \#lvl = 2 \qquad \#ge\text{-}sp = 5 \qquad \#nc\text{-}cl = 3$$

Résultat des conversions La figure 5.2 montre 3 modèles, chacun représentant un résultat de conversion d'héritage pour un niveau de représentation différent.
Le premier modèle (sous-figure 5.2a) permet une représentation au niveau le plus général. La classe choisie pour chaque feuille de la hiérarchie (i.e. $\mathtt{C}_{Thesis}$, $\mathtt{C}_{InProceedings}$ et $\mathtt{C}_{JournalArticle}$) est la classe $\mathtt{C}_{Document}$. Ainsi, toutes les classes ont été fusionnées dans la plus haute classe de la hiérarchie, i.e. $\mathtt{C}_{Document}$. Elle possède désormais tous les attributs de la hiérarchie et toutes les instances sont devenues des instances de $\mathtt{C}_{Document}$. L'appartenance à une précédente classe (e.g. $\mathtt{C}_{Report}$) est représentée avec un des 4 nouveaux attributs produits à partir de ces classes (e.g. $\mathtt{a}_{Report}$).

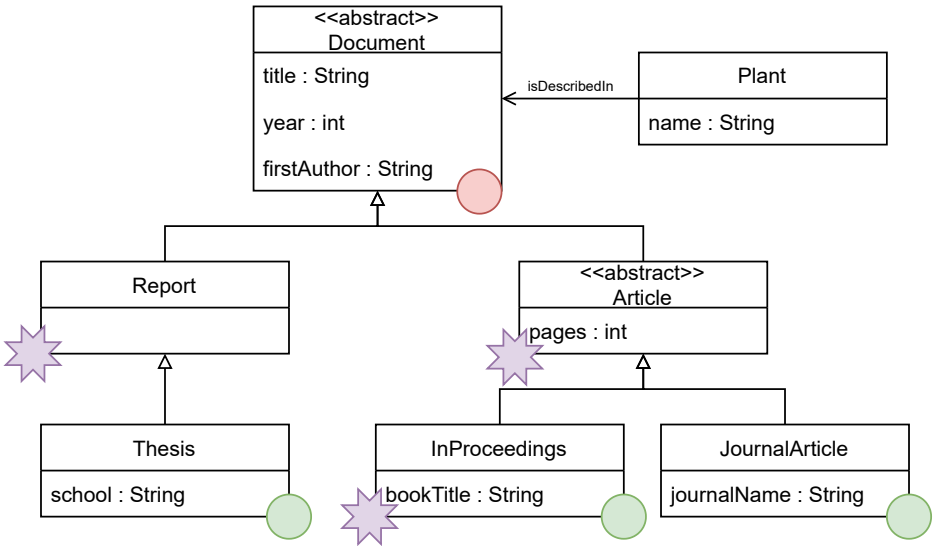
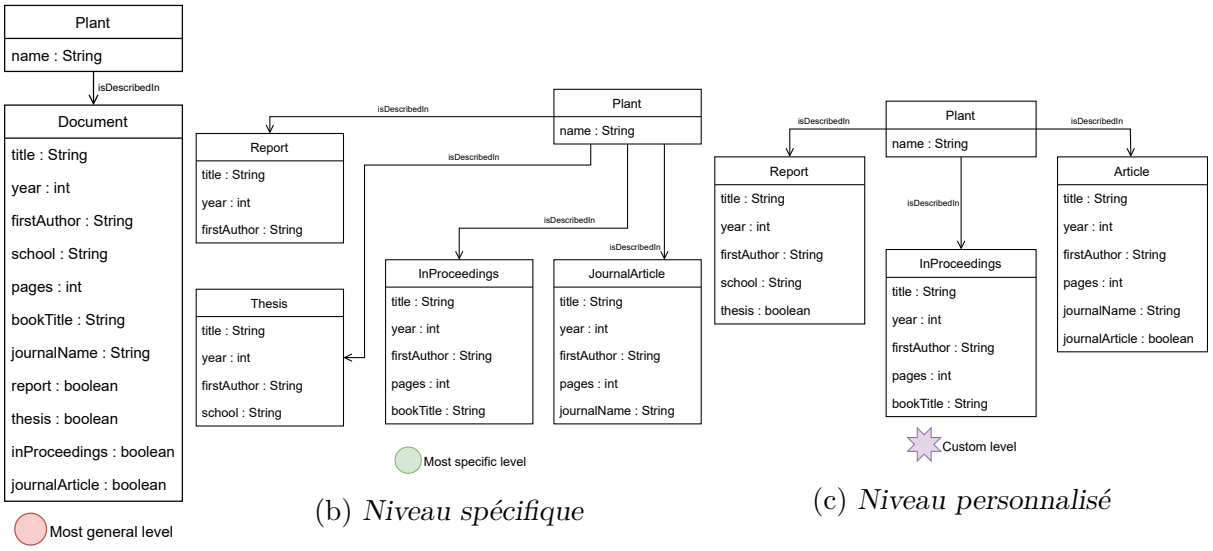


FIGURE 5.1 – *Modèle de données focalisé sur les références bibliographiques de Knomana. Dans ce modèle, une plante est décrite dans un document. Les documents peuvent être des rapports, et plus précisément des thèses, ou alors des articles de conférence ou de journal. Les pastilles représentent les classes sélectionnées pour chaque niveau de représentation sélectionné dans la suite de cette sous-section.*

Le deuxième modèle (sous-figure 5.2b) permet une représentation au niveau le plus spécifique. La classe choisie pour chaque feuille de la hiérarchie (i.e. $\mathtt{C}_{Thesis}$, $\mathtt{C}_{InProceedings}$ et $\mathtt{C}_{JournalArticle}$) est la feuille elle-même. Ainsi, toutes les classes concrètes du modèle original sont donc conservées et tous les attributs possédés par chacune d'entre elles sont explicitement notés. De même, l'association $\mathtt{r}_{isDescribedIn}$ entre $\mathtt{C}_{Plant}$ et l'ancienne $\mathtt{C}_{Document}$ est dupliquée pour chacune des sous-classes restantes.
Le troisième modèle (sous-figure 5.2c) permet une représentation à un niveau personnalisé. Une classe différente a été choisie pour chaque branche de la hiérarchie : $\mathtt{C}_{Report}$ pour la feuille $\mathtt{C}_{Thesis}$, $\mathtt{C}_{InProceedings}$ pour la feuille $\mathtt{C}_{InProceedings}$ et $\mathtt{C}_{Article}$ pour la feuille $\mathtt{C}_{JournalArticle}$. Ainsi, $\mathtt{C}_{Thesis}$ a été fusionnée dans $\mathtt{C}_{Report}$. $\mathtt{C}_{Report}$ possède donc deux nouveaux attributs en plus de ceux inhérents à la hiérarchie : $\mathtt{a}_{school}$, l'attribut de $\mathtt{C}_{Thesis}$ et $\mathtt{a}_{thesis}$, l'attribut produit à partir de $\mathtt{C}_{Thesis}$. Il en va de même pour $\mathtt{C}_{JournalArticle}$ qui a été fusionnée dans $\mathtt{C}_{Article}$, et ses attributs qui ont été ajoutés dans cette classe. L'association $\mathtt{r}_{isDescribedIn}$ est là aussi dupliquée pour chacune des sous-classes de $\mathtt{C}_{Document}$ restantes.

Pour transformer les modèles de la figure 5.2 en modèles de type UML_{CoBUBo}, chaque attribut multi-valué est converti par un *scaling*. Un *scaling* nominal est appliqué pour les attributs $\mathtt{a}_{title}$, $\mathtt{a}_{author}$, $\mathtt{a}_{school}$, $\mathtt{a}_{bookTitle}$ et $\mathtt{a}_{journalName}$. L'attribut $\mathtt{a}_{year}$ est converti par un *scaling* ordinal. Enfin, $\mathtt{a}_{pages}$ est converti avec notre *scaling* par intervalles avec les intervalles suivants : [0,6], [6,16], [10,40] et [20,500], avec union des intervalles voisins à chaque étage comme illustré dans la table 5.1 (dans laquelle les nombres de la colonne de gauche ne servent qu'à illustrer le *scaling*).

Résultat du passage à l'échelle Une fois les attributs multi-valués transformés en ensembles d'attributs binaires, on obtient des RCFs (une par modèle de la figure 5.2) dont les dimensions sont présentées dans la table 5.2. Cette table présente, pour chacun des 3



(a) *Niveau général*

(b) *Niveau spécifique*

(c) *Niveau personnalisé*

FIGURE 5.2 – *Conversion du modèle de la figure 5.1. Chaque sous-figure représente le résultat produit à un niveau de représentation différent. Les pastilles font écho à celles présentes dans la figure 5.1.*

pages	[0,6]	[6,16]	[10,40]	[20,500]	[0,16]	[6,40]	[10,500]	[0,40]	[6,500]	[0,500]
4	×				×			×		×
6	×	×			×	×		×	×	×
11		×	×		×	×	×	×	×	×
18			×			×	×	×	×	×
72				×			×		×	×

TABLE 5.1 – *Scaling par intervalle appliqué pour le nombre de pages de* $\mathtt{C}_{Article}$*. Les nombres dans la colonne de gauche ne servent qu'à illustrer le scaling appliqué*

.

modèles, le nombre total d'objets et d'attributs de la RCF obtenue après la conversion. On y voit également la répartition des attributs binaires produits pour chacun des attributs multi-valués. Pour le niveau général (figure 5.2a), le nombre d'objets du contexte formel $\mathcal{K}_{Document}$ correspond au nombre de documents contenus dans le jeu de données (788). Pour le niveau spécifique (figure 5.2b), le nombre total de documents présents dans les différents contextes formels est de 789. Cela est dû au fait que l'objet présent dans $\mathcal{K}_{Thesis}$ l'est également dans $\mathcal{K}_{Report}$ grâce à la duplication. Pour le niveau personnalisé (figure 5.2c), le nombre total de documents présents dans les différents contextes formels est de 810. En effet, tous les objets de $\mathcal{K}_{InProceedings}$ sont également présents dans $\mathcal{K}_{Article}$. Le même principe s'applique pour les nombres d'attributs présents dans les contextes formels des différents niveaux [4].

4. Les nombres de valeurs pour $\mathtt{a}_{firstAuthor}$ et $\mathtt{a}_{journalName}$ sont différents de ceux du jeu de données (respectivement 642 pour 647 et 302 pour 303) à cause de l'absence d'une tâche de pré-traitement (sur le fichier CSV) pour homogénéiser les valeurs similaires (accents, espace, etc.).

				#values for each attribute										
Generalisation Level	Formal context (Class)	#objects	#attributes	#title	#year	#firstAuthor	#school	#pages	#bookTitle	#journalName	#report	#thesis	#inProceedings	#journalArticle
General	Document	788	1751	778	4	642	2	10	5	302	2	2	2	2
Specific	Report	7	18	7	4	7								
	Thesis	1	7	1	4	1	1							
	InProceedings	22	62	22	4	21		10	5					
	JournalArticle	759	1685	749	4	620		10		302				
Custom	Report	7	18	7	4	7								
	InProceedings	22	62	22	4	21			5					
	Article	781	1724	771	4	635		10		302				2
All	Plant	3096	2915											

TABLE 5.2 – *Nombre d'objets et d'attributs binaires des contextes formels après conversion. Les onze dernières colonnes représentent la répartition des attributs binaires en fonction des attributs multi-valués correspondants.*

5.2.2 Évaluation quantitative de la conversion d'une association n-aire

La deuxième partie de l'algorithme de conversion concerne la conversion des associations. Les deux cas d'étude suivants sont notamment focalisés sur la conversion d'une relation n-aire.

Cas d'étude n° 1 sur des systèmes de protection (Data11P) Ce jeu de données concerne 11 plantes de protection, i.e. *Azadirachta indica*, *Calotropis procera*, *Cymbopogon citratus*, *Dysphania ambrosioides*, *Hyptis suaveolens*, *Lantana camara*, *Moringa oleifera*, *Ocimum gratissimum*, *Securidaca longipedunculata*, *Tephrosia vogelii*, et *Thymus vulgaris*. Il comprend un total de 413 descriptions d'usages de plante composées de 11 attributs multi-valués : 3 pour la plante utilisée (nom de l'espèce, famille et origine), 4 pour l'organisme ciblé (nom, genre famille et règne) et 4 pour l'organisme protégé (nom, genre, famille et règne). Pour ce cas d'étude, afin d'avoir une représentation un peu plus fidèle des biopesticides, les instances de la classe $\mathtt{C}_{UsedPlant}$ sont formées par la combinaison de l'espèce et du territoire, ce qui forme un total de 73 instances.

Le modèle illustré par la Figure 5.3 est focalisé sur l'association ternaire entre les 3 éléments principaux de PPAf. La plante de protection ($\mathtt{C}_{UsedPlant}$) est décrite par son nom, sa famille et son lieu de provenance. L'organisme ciblé et l'organisme protégé (respectivement $\mathtt{C}_{TargetedOrganism}$ et $\mathtt{C}_{ProtectedOrganism}$) sont uniquement décrits par des attributs issus de la classification taxonomique : leur nom, leur genre, leur famille et leur règne.

Résultat de l'applicabilité Data11P contient 1 association n-aire entre les biopesticides (*#n-aire*), les organismes protégés et les organismes ciblés. Cette association concerne respectivement 11 plantes de protection originaires de plusieurs pays, 30 organismes à protéger et 150 organismes ciblés pour un total de 398 liens entre ces entités.

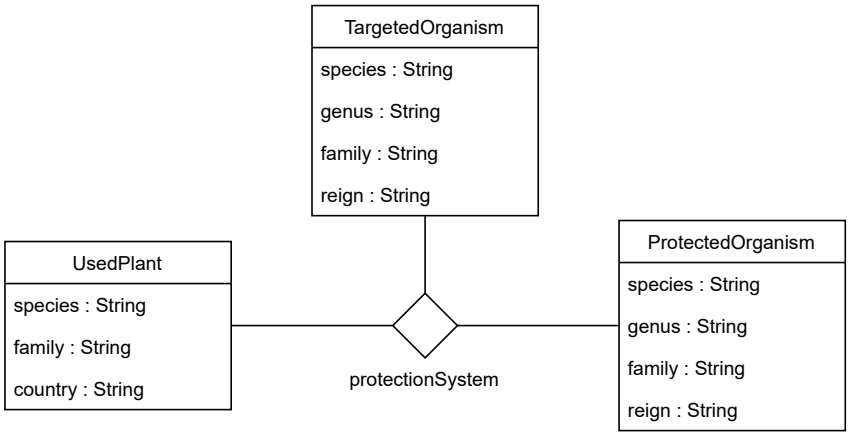


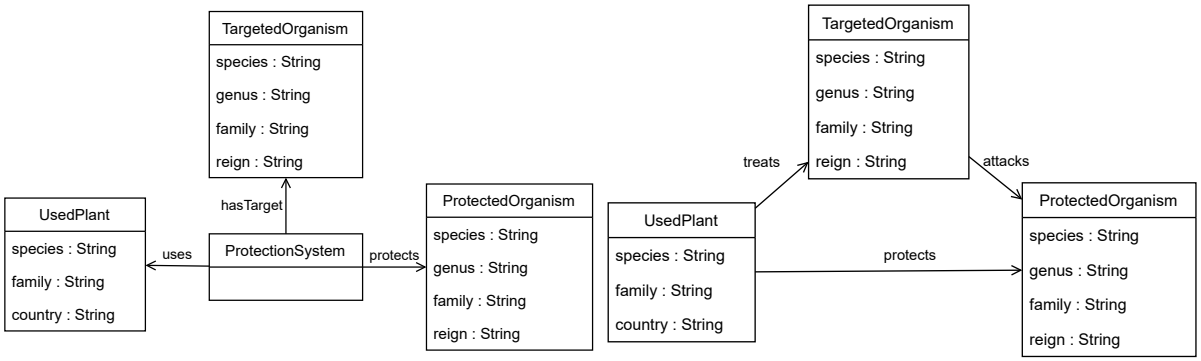
FIGURE 5.3 – *Modèle de données focalisé sur des systèmes de protection décrits dans Knomana.*

Ainsi, même si les opérations $\mathrm{Na}_{\mathrm{split}}$ ou $\mathrm{Na}_{\mathrm{mat}}$ ne sont appliquées qu'une seule fois, pour seulement 73 instances de $\mathtt{C}_{UsedPlant}$, cela concerne une quantité importante de liens (environ 400 liens). Pour résumer :

$$\#n\text{-}ary = 1 \qquad \#tri = 398$$

Résultat des conversions La figure 5.4 montre 2 modèles, chacun représentant un résultat soit de $\mathrm{Na}_{\mathrm{mat}}$ soit de $\mathrm{Na}_{\mathrm{split}}$.
La figure de gauche (sous-figure 5.4a) représente le modèle obtenu en appliquant, entre autres, l'opération $\mathrm{Na}_{\mathrm{mat}}$. L'association n-aire $\mathtt{r}_{protectionSystem}$ a été matérialisée en une nouvelle classe avec $\mathrm{Na}_{\mathrm{mat}}$. Cela a également produit 3 associations binaires bidirectionnelles pour relier la nouvelle classe $\mathtt{C}_{ProtectionSystem}$ aux 3 autres classes. Chacune de ces associations bidirectionnelles a été transformée en 2 associations unidirectionnelles avec Bi_{2uni}. Finalement, seul un sens a été conservé pour chaque paire d'associations unidirectionnelles dans le modèle final présenté par la figure.
La figure de droite (sous-figure 5.4b) représente le modèle obtenu en appliquant, entre autres, l'opération $\mathrm{Na}_{\mathrm{split}}$. L'association n-aire $\mathtt{r}_{protectionSystem}$ a été divisée en 3 associations binaires bidirectionnelles avec $\mathrm{Na}_{\mathrm{split}}$ afin de relier chacune des 3 classes avec toutes les autres. Chacune de ces associations bidirectionnelles a été transformée en 2 associations unidirectionnelles avec Bi_{2uni}. Finalement, seul un sens a été conservé pour chaque paire d'associations unidirectionnelles dans le modèle final présenté par la figure. Pour transformer les modèles de la figure 5.4 en modèle de type UML_{CoBUBo}, chaque attribut multi-valué est converti par un *scaling*. Tous les attributs sont de type nominal, ils sont donc tous convertis grâce à un *scaling* nominal.

Résultat du passage à l'échelle Une fois les attributs multi-valués transformés en ensembles d'attributs binaires, on obtient des RCFs (une par modèle de la figure 5.4) dont les dimensions sont présentées dans la table 5.3. Cette table présente le nombre total d'objets et d'attributs par contexte formel obtenu après la conversion. En utilisant l'opération $\mathrm{Na}_{\mathrm{mat}}$, le nombre d'objets de $\mathcal{K}_{ProtectionSystem}$ représente le nombre total de descriptions du jeu de données. En utilisant l'opération $\mathrm{Na}_{\mathrm{split}}$, le nombre total d'objets est inférieur au nombre total de descriptions du jeu de données mais la signification



(a) *Modèle obtenu avec l'application notamment de* Na_{mat}

(b) *Modèle obtenu avec l'application notamment de* Na_{split}

FIGURE 5.4 – *Conversion du modèle de la figure 5.3. Chaque sous-figure représente le résultat produit respectivement par l'application de* Na_{mat} *(à gauche) et* Na_{split} *(à droite).*

sous-jacente à la combinaison des relations utilisées ne sera pas la même (au profit de la génération d'hypothèses) [5].

Formal context	#objects	#attributes	Multi-valued attributes	#values
ProtectionSystem (only Na_{mat})	372	0	-	-
UsedPlant	73	47	species	11
			family	9
			country	27
ProtectedOrganism	30	64	species	30
			genus	19
			family	12
			reign	3
TargetedOrganism	150	314	species	150
			genus	103
			family	57
			reign	4

TABLE 5.3 – *Nombre d'objets et d'attributs binaires des contextes formels après conversion. Les deux dernières colonnes représentent la répartition des attributs binaires en fonction des attributs multi-valués correspondants.*

Cas d'étude n° 2 sur des systèmes de protection (Data6P) Ce jeu de données concerne 6 plantes de protection, i.e. *Cymbopogon citratus*, *Hyptis suaveolens*, *Lantana camara*, *Moringa oleifera*, *Ocimum gratissimum*, et *Thymus vulgaris*. Il comprend un total de 225 descriptions d'usages de plante composées de 16 attributs multi-valués : 3 pour la plante utilisée (nom de l'espèce, genre et famille), 3 pour son origine (territoire, partie du continent et continent), 4 pour l'organisme ciblé (nom, genre famille et règne), 4 pour l'organisme protégé (nom, genre, famille et règne), et 2 pour le système protégé (1 pour l'étape de production e.g. culture ou bétail, et 1 pour le lieu du traitement e.g. sur le terrain ou dans les stocks).

5. Les nombres d'objet et d'attributs obtenus peuvent ne pas correspondre au jeu de données original. C'est notamment le cas pour le nombre d'objets de $\mathcal{K}_{ProtectionSystem}$ (pour l'application de Na_{mat}) qui ne correspond pas au nombre total de descriptions d'usages présentes dans Data11P (resp. 372 objets pour 413 descriptions). Ces différences sont dues au traitement de certaines valeurs présentes dans le fichier original, e.g. #N/A.

Si le jeu de données contient moins de descriptions que dans le cas précédent (Data11P), le modèle est cependant un peu plus élaboré grâce à la présence de classes supplémentaires autour des 3 éléments principaux. Le modèle illustré par la figure 5.5 est focalisé sur l'association ternaire entre les 3 éléments principaux de PPAf. Le premier est le biopesticide ($\mathtt{C}_{BioPesticide}$) fabriqué à partir d'une plante ($\mathtt{C}_{UsedPlant}$) originaire d'un lieu ($\mathtt{C}_{Location}$). Le deuxième est l'organisme ciblé ($\mathtt{C}_{TargetedOrganism}$). Et le troisième est le système protégé ($\mathtt{C}_{ProtectedSystem}$) concernant un organisme ($\mathtt{C}_{ProtectedOrganism}$). Les 3 organismes sont décrits par des attributs issus de la classification taxonomique (espèce, genre, etc.)

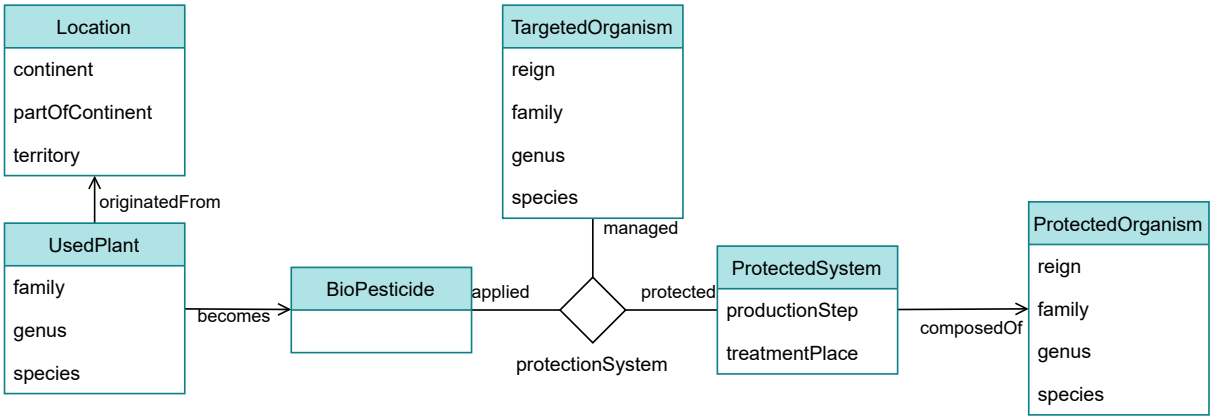


FIGURE 5.5 – *Modèle de données focalisé sur des systèmes de protection décrits dans Knomana. Comparé au précédent, 3 classes sont ajoutées. La première matérialise les biopesticides ($\mathtt{C}_{Biopesticide}$). La deuxième ($\mathtt{C}_{Location}$) regroupe tout ce qui concerne la location. Et la dernière ($\mathtt{C}_{ProtectedSystem}$) donne plus d'informations sur les systèmes protégés.*

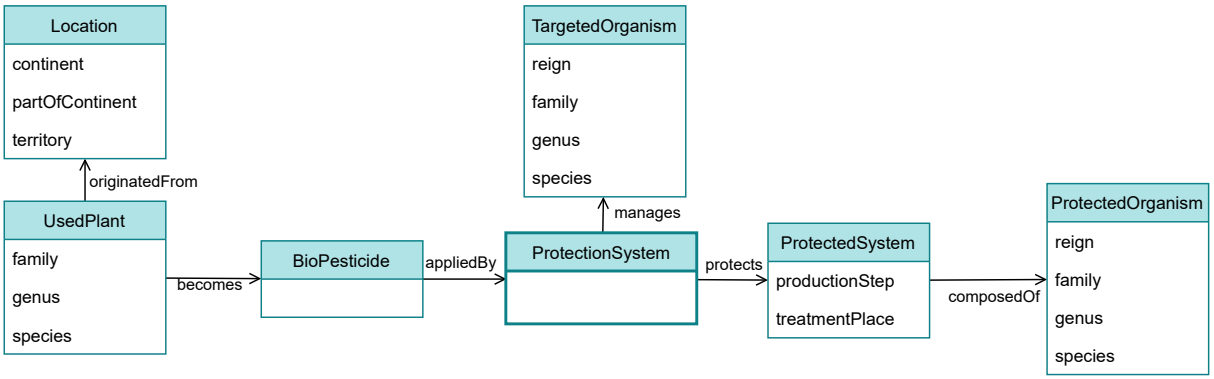
.

Résultat de l'applicabilité Data6P contient 1 association n-aire (*#n-aire*) entre les biopesticides, les systèmes protégés et les organismes ciblés. Cette association concerne respectivement 38 biopesticides (pour 6 plantes de protection), 14 systèmes protégés (pour 21 organismes à protéger) et 111 organismes ciblés pour un total de 225 liens entre ces entités. En synthèse :

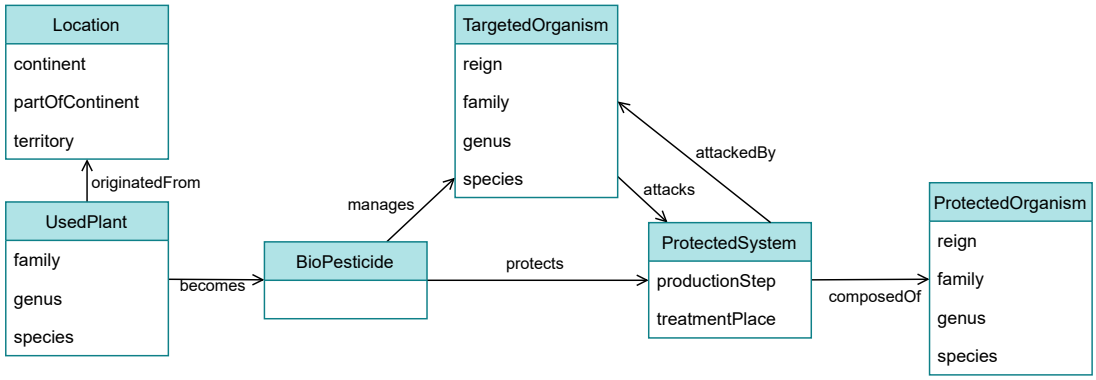
$$\#n\text{-}ary = 1 \qquad \#tri = 225$$

Résultat des conversions La figure 5.6 montre 2 modèles, chacun représentant un résultat impliquant une conversion d'association n-aire.
La figure du haut (sous-figure 5.6a) représente le modèle obtenu en appliquant, entre autres, l'opération $\mathrm{Na}_{\mathrm{mat}}$. L'association n-aire $\mathtt{r}_{protectionSystem}$ a été matérialisée en une nouvelle classe. Cela a également produit 3 associations binaires bidirectionnelles pour relier la nouvelle classe $\mathtt{C}_{ProtectionSystem}$ aux 3 autres classes. Chacune de ces associations bidirectionnelles a été transformée en 2 associations unidirectionnelles avec $Bi_{2\mathrm{uni}}$. Finalement, seul un sens a été conservé pour chaque paire d'associations unidirectionnelles dans le modèle final présenté par la figure.
La figure du bas (sous-figure 5.6b) représente le modèle obtenu en appliquant, entre autres, l'opération $\mathrm{Na}_{\mathrm{split}}$. L'association n-aire $\mathtt{r}_{protectionSystem}$ a été divisée en 3 associations binaires bidirectionnelles avec $\mathrm{Na}_{\mathrm{split}}$ afin de relier chacune des 3 classes avec toutes les

autres. Chacune de ces associations bidirectionnelles a été transformée en 2 associations unidirectionnelles avec $Bi_{2\text{uni}}$. Finalement, seul un sens a été conservé pour deux des 3 paires d'associations unidirectionnelles dans le modèle final présenté par la figure.



(a) *Modèle obtenu avec* Na_{mat}



(b) *Modèle obtenu avec* Na_{split}

FIGURE 5.6 – *Conversion du modèle de la figure 5.5. Chaque sous-figure représente le résultat produit respectivement par l'application de* Na_{mat} *(en haut) et* Na_{split} *(en bas)*

Pour passer les modèles de la figure 5.6 en modèle de type UML_{CoBUBo}, chaque attribut multi-valué est converti par un *scaling* nominal dans la mesure où tous les attributs sont de type nominal.

Résultat du passage à l'échelle Une fois les attributs multi-valués transformés en ensembles d'attributs binaires, on obtient des RCFs (une par modèle de la figure 5.6) dont les dimensions sont présentées dans la table 5.4. Cette table présente le nombre total d'objets et d'attributs par contexte formel obtenu après la conversion. En utilisant la transformation Na_{mat}, le nombre d'objets de $\mathcal{K}_{ProtectionSystem}$ représente le nombre total de descriptions du jeu de données. En utilisant l'opération Na_{split}, le nombre total d'objets est inférieur au nombre total de descriptions du jeu de données mais la signification sous-jacente à la combinaison des relations utilisées ne sera pas la même (au profit de la génération d'hypothèses).

Formal context (Class name)	#objects	#attributes	Included data types	#values
BioPesticide	38	0	(empty class)	-
UsedPlant	6	16	family	4
			genus	6
			species	6
Location	20	32	continent	5
			partOfContinent	7
			territory	20
ProtectedSystem	14	19	productionStep	7
			treatmentPlace	12
ProtectedOrganism	21	48	reign	3
			family	11
			genus	15
			species	19
TargetedOrganism	111	234	reign	5
			family	43
			genus	78
			species	108
ProtectionSystem	225	0	ternary relation	-

TABLE 5.4 – *Nombre d'objets et d'attributs binaires des contextes formels après conversion. Les deux dernières colonnes représentent la répartition des attributs binaires en fonction des attributs multi-valués correspondants.*

5.2.3 Évaluation quantitative pour une classe d'association

Le cas d'étude suivant concerne l'ensemble du jeu de connaissances de PPAf recensées en Octobre 2019. Pour cette évaluation, les connaissances n'ont reçu aucun pré-traitement, il s'agit donc des connaissances brutes. Grâce à son modèle de données plus complet et à la présence de toutes les connaissances, il permet une évaluation quantitative de l'impact de la conversion sur la taille d'une RCF sur un jeu volumineux.

Cas d'étude sur le jeu complet de PPAf Ce jeu de données est constitué de l'ensemble des connaissances contenues dans PPAf en Octobre 2019. Il rassemble 40 800 descriptions d'usages de plante décrites par 23 attributs multi-valués. Ces descriptions concernent 523 espèces de plante, 127 organismes ciblés et 28 organismes protégés.

Le modèle illustré par la figure 5.7 représente l'organisation des principales informations contenues dans PPAf en Octobre 2019. L'élément central de ce modèle est l'association n-aire qui relie quatre éléments : les connaissances d'un document, le système protégé, l'organisme ciblé ainsi que le produit utilisé. Chaque document ($\mathtt{C}_{Document}$) est décrit par toutes les informations disponibles parmi lesquelles : le titre, l'année, le nom du premier auteur, le DOI, s'il s'agit d'une *review*, etc. Il possède des connaissances ($\mathtt{C}_{Knowledge}$) qui décrivent des relations de protection sanitaire. Le système protégé ($\mathtt{C}_{ProtectedSystem}$) représente un organisme ($\mathtt{C}_{ProtectedOrganism}$) sur lequel est appliquée la méthode de protection. L'organisme ciblé est un ravageur que l'on souhaite contrôler ($\mathtt{C}_{TargetedOrganism}$). Enfin, le biopesticide utilisé ($\mathtt{C}_{Biopesticide}$) est fabriqué à partir d'une plante de protection ($\mathtt{C}_{UsedPlant}$) dont on extrait une ou plusieurs parties ($\mathtt{C}_{PlantPart}$).

Résultat de l'applicabilité Ce jeu de données contient 1 association n-aire (*#n-aire*) entre les documents, les biopesticides, les systèmes protégés et les organismes ciblés.

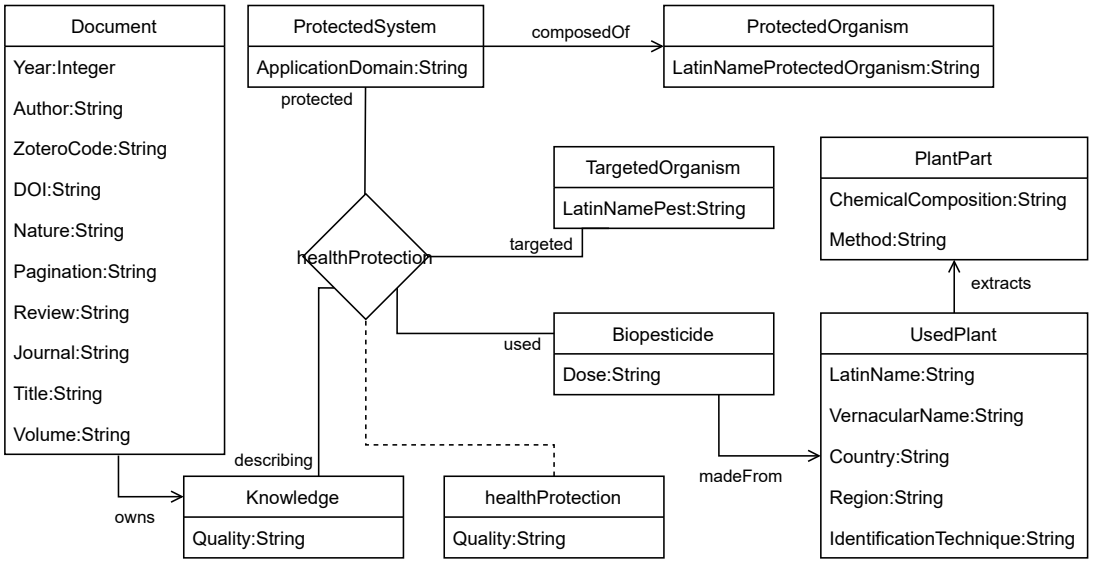


FIGURE 5.7 – *Modèle représentatif de Knomana. Un document contient une connaissance qui décrit un système de protection sanitaire. Cette protection sanitaire est composée du document, du système protégé, de l'organisme ciblé ainsi que du produit utilisé.*

Cette association concerne respectivement 3 541 documents, 3 127 biopesticides (pour 4 078 plantes de protection), 6 systèmes protégés (pour 195 organismes à protéger) et 934 organismes ciblés pour un total de 10 172 liens entre ces entités. En synthèse :

$$\#n\text{-}ary = 1 \qquad \#tri = 10172$$

Résultat de la conversion La figure 5.8 montre le modèle après conversion. L'association n-aire étant également une classe d'association, elle est d'abord matérialisée en utilisant Att_{mat}. L'association n-aire $\mathtt{r}_{healthProtection}$ disparaît donc au profit d'une nouvelle classe $\mathtt{C}_{HealthProtection}$ et de 4 associations binaires bidirectionnelles pour relier $\mathtt{C}_{HealthProtection}$ aux 4 autres classes. Chacune de ces associations bidirectionnelles a été transformée en 2 associations unidirectionnelles avec $Bi_{2\text{uni}}$. Finalement, seul un sens a été conservé pour chaque paire d'associations unidirectionnelles dans le modèle final présenté par la figure.

Pour transformer le modèle de la figure 5.8 en modèle de type UML_{CoBUBo}, chaque attribut multi-valué est converti par un *scaling*. Le but étant d'avoir le volume le plus large possible, tous les attributs multi-valués ont été convertis grâce à un *scaling* nominal.

Résultat du passage à l'échelle Une fois les attributs multi-valués transformés en ensembles d'attributs binaires, on obtient une RCF dont les dimensions sont présentées dans la table 5.6. Cette table présente le nombre total d'objets et d'attributs par contexte formel obtenu après la conversion (originaux et au final obtenus avec les *scaling*). La table contient également la densité de chacun des contextes formels et relationnels. La densité est au maximum de l'ordre de 16% pour les contextes formels et de 20% pour les contextes relationnels. Comme la classe d'association a été matérialisée avec l'opération Att_{mat}, le nombre d'objets de $\mathcal{K}_{HealthProtection}$ représente le nombre total de descriptions du jeu de données.

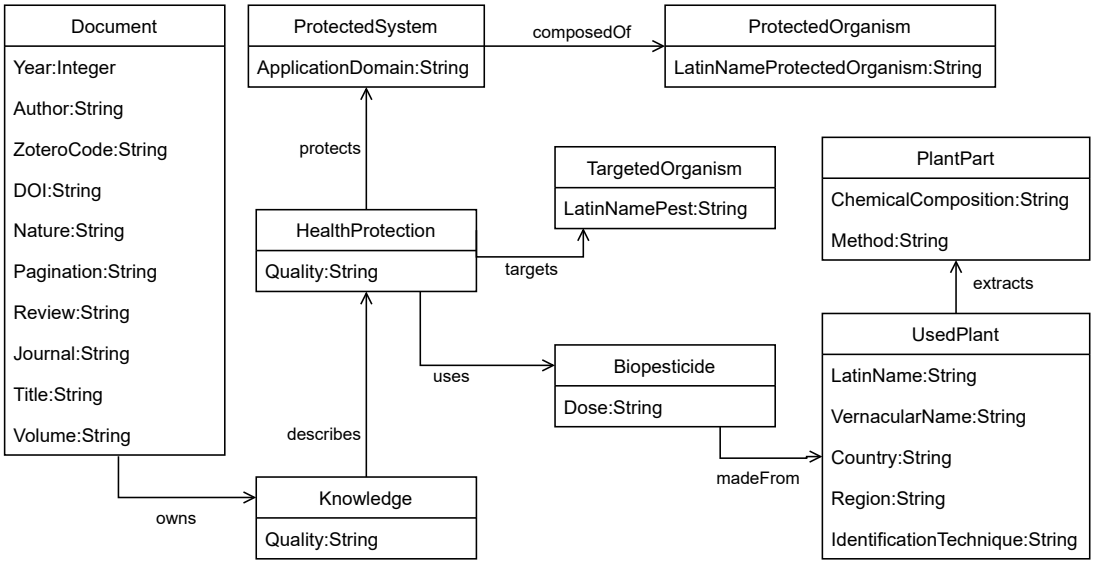


FIGURE 5.8 – *Conversion du modèle de la figure 5.7 grâce à l'opération* Att_{mat}.

5.3 Évaluation des structures conceptuelles

La section précédente présente les résultats de la conversion de jeux de données en RCFs. Ces RCFs servent de données d'entrée pour les méthodes de classification, i.e. l'Analyse de Concepts Formels ou son extension relationnelle. L'objet de cette section est de montrer la pertinence de la méthode de conversion sur les structures conceptuelles obtenues. L'analyse quantitative est effectuée pour les cas d'étude DataDoc et Data11P, et l'analyse qualitative sur le cas d'étude Data6P.

Cette section présente tout d'abord une évaluation quantitative de ces structures conceptuelles incluant de l'héritage et une relation n-aire (section 5.3.1), suivie d'une évaluation qualitative incluant une relation n-aire (section 5.3.2). Enfin, les structures conceptuelles produites par FCA et diverses extensions de FCA sont évaluées qualitativement sur les données indéterminées.

5.3.1 Évaluation quantitative

Dans cette section, des indicateurs complémentaires sont utilisés pour évaluer les structures conceptuelles produites avec les RCFs précédemment obtenues. La mesure du passage à l'échelle se fait à l'aide de 4 indicateurs sur les structures conceptuelles finales : le nombre de concepts pour chaque treillis (*#concepts*), le nombre d'attributs relationnels pour chaque treillis (*#rel-att*), le temps de calcul (*#msec*) et le nombre d'étapes (*#steps*).

Chaque indicateur est calculé pour les 4 types de structures conceptuelles produites avec 6 algorithmes dédiés. Les treillis de concepts (LAT) sont produits en utilisant l'algorithme ADD-INTENT [Merwe et al., 2004]. Les AOC-posets (AOC) sont produits en utilisant les algorithmes ARES, CERES, HERMES et PLUTON [Berry et al., 2014] qui ont des temps

Formal context	#Objects	#original attributes	#Boolean attributes	density	**Relational context**	density
Document	3541	10	6350	7.73 E-4	**owns**	0.062
Knowledge	16	1	16	0.062	**describes**	0.063
HealthProtection	10172	1	30	0.033	**protects**	0.168
ProtectedSystem	6	1	6	0.167	**composedOf**	0.206
ProtectedOrganism	195	1	195	0.005		
					targets	0.001
TargetedOrganism	934	1	934	0.001		
					uses	3.2 E-4
Biopesticide	3127	1	30	0.033	**madeFrom**	3.23 E-4
UsedPlant	4078	5	4353	7.37 E-4	**extracts**	0.068
PlantPart	344	2	377	0.005		

TABLE 5.5 – *Knomana: Dimensions of the relational context family*

TABLE 5.6 – *Nombre d'objets et d'attributs (multi-valués puis booléens) des contextes formels après conversion. La densité de tous les contextes formels et relationnels de la RCF est également indiquée.*

d'exécution différents. Les treillis Icebergs avec des seuils de 30% et 40% (ICE40 et ICE40) sont produits en utilisant l'algorithme TITANIC [Stumme et al., 2002].

Dans cette section, les RCFs résultant des conversions (voir section 5.2) sont appelées « modèles unidirectionnels ». L'appellation « modèles bidirectionnels » est utilisée pour mentionner les résultats des conversions pour lesquelles toutes les relations possèderaient leur relation inverse (quand elle n'est pas déjà présente).

Structures conceptuelles incluant l'héritage (DataDoc)

Les tables 5.7 et 5.8 présentent respectivement les nombres de concepts et d'attributs relationnels pour chaque structure conceptuelle. Pour les modèles unidirectionnels, le nombre total de concepts et d'attributs relationnels varie respectivement de 27 à 16 283 concepts et de 19 à 1481 attributs relationnels. Pour les modèles bidirectionnels, ces nombres varient de 54 à 7 577 concepts et de 120 à 22 311 attributs. La notation « oom » signifie « manque de mémoire ». En effet, pour les modèles bidirectionnels, les treillis de concepts n'ont pu être construits à cause du manque de mémoire sur l'ordinateur. L'utilisation de modèles bidirectionnels augmente au moins de cinq fois le nombre de concepts par rapport à l'utilisation de modèles unidirectionnels. Cela peut aller jusqu'à 35 fois dans le cas des treillis Icebergs avec le niveau personnalisé et 22 fois pour les AOC-posets avec le niveau spécifique. L'application du niveau spécifique produit moins de concepts et d'attributs relationnels que les deux autres modèles.

	General level				Specific level				Custom level			
	LAT	AOC	ICE30	ICE40	LAT	AOC	ICE30	ICE40	LAT	AOC	ICE30	ICE40
	unidirectional models											
Plant	14802	3525	19	15	13771	3520	8	8	14575	3522	15	11
Document	1481	982	31	25								
Report					12	10	5	5	12	10	5	5
Thesis					2	2	2	2				
InProceeding					40	32	4	4	40	32	4	4
JournalArticle					1348	937	9	8				
Article									1415	967	19	15
TOTAL	16283	4507	50	40	15173	4501	28	27	16042	4531	43	35
	bidirectional models											
Plant	oom	4936	268	44	oom	4933	33	22	oom	4952	263	39
Document	oom	2569	381	83								
Report					oom	11	6	6	oom	11	6	6
Thesis					oom	2	2	2				
InProceeding					oom	51	6	6	oom	52	6	6
JournalArticle					oom	2515	30	18				
Article									oom	2562	188	34
TOTAL	oom	7505	649	127	oom	7512	77	54	oom	7577	463	85

TABLE 5.7 – *Nombre de concepts (#concepts) par contexte formel produits à partir des résultats du cas d'étude DataDoc pour chaque structure conceptuelle : treillis de concepts* (LAT)*, AOC-posets* (AOC) *et treillis Iceberg* (ICE30) *et* (ICE40)*. « oom » signifie « out of memory ».*

	General level				Specific level				Custom level			
	LAT	AOC	ICE30	ICE40	LAT	AOC	ICE30	ICE40	LAT	AOC	ICE30	ICE40
	unidirectional models											
Plant	1481	982	31	25	1402	981	20	19	1467	1009	28	24
Document	0	0	0	0								
Report					0	0	0	0	0	0	0	0
Thesis					0	0	0	0				
InProceeding					0	0	0	0	0	0	0	0
JournalArticle					0	0	0	0				
Article									0	0	0	0
TOTAL	1481	982	31	25	1402	981	20	19	1467	1009	28	24
	bidirectional models											
Plant	oom	2569	381	83	oom	2579	44	32	oom	2625	200	46
Document	oom	4936	268	44								
Report					oom	4933	33	22	oom	4952	263	39
Thesis					oom	4933	33	22				
InProceeding					oom	4933	33	22	oom	4952	263	39
JournalArticle					oom	4933	33	22				
Article									oom	4952	263	39
TOTAL	oom	7505	649	127	oom	22311	176	120	oom	17481	989	163

TABLE 5.8 – *Nombre d'attributs relationnels (#rel-att) par contexte formel produits à partir des résultats du cas d'étude DataDoc pour chaque structure conceptuelle : treillis de concepts* (LAT)*, AOC-posets* (AOC) *et treillis Iceberg* (ICE30) *et* (ICE40)*. « oom » signifie « out of memory ».*

L'ensemble de données correspondant concerne 788 instances de documents (*#doc*). Au niveau général, ces liens sont représentés dans le contexte formel $\mathcal{K}_{Document}$. Dans le niveau spécifique, ils sont représentés par les contextes formels $\mathcal{K}_{Report}$, $\mathcal{K}_{InProceedings}$ et $\mathcal{K}_{JournalArticle}$. $\mathtt{C}_{Thesis}$ étant une sous-classe de $\mathtt{C}_{Report}$, tous les documents concernés par $\mathcal{K}_{Thesis}$ sont représentés dans $\mathcal{K}_{Report}$. Au niveau personnalisé, ils sont représentés dans les contextes formels $\mathcal{K}_{Report}$ et $\mathcal{K}_{Article}$. Pour la même raison que précédemment, tous les documents concernés par $\mathcal{K}_{JournalArticle}$ sont représentés dans $\mathcal{K}_{Article}$. Pour les modèles unidirectionnels, concernant ces nombres, les treillis de concepts sont environ 2 fois plus grands (1 481 pour le niveau général, 1 400 pour le niveau spécifique et 1 427 pour le niveau personnalisé) et les treillis Icebergs peuvent être trop petits pour donner des informations pertinentes (environ 20 concepts), tandis que les AOC-posets sont plus acceptables avec 982, 979 et 977 concepts. Pour les modèles bidirectionnels, les treillis de concepts sont

trop grands pour être calculés et les AOC-posets sont 3 fois plus grands que *#doc*. Les treillis Icebergs sont plus acceptables mais contiennent moins d'informations en raison du seuil minimal de support.

Comme illustré dans la table 5.8, si une classe du diagramme UML_{CoBUBo} (le diagramme de classes UML final avant conversion) est un puits, le contexte formel correspondant $\mathcal{SFC}$ ne possède aucun attribut relationnel. La valeur de *#concepts* pour $\mathcal{SFC}$ est directement cumulée dans la valeur *#rel-att* du contexte formel qui est le domaine de r. Pour les modèles bidirectionnels, les valeurs de l'indicateur *#rel-att* de $\mathcal{K}_{Document}$, $\mathcal{K}_{Report}$, $\mathcal{K}_{Thesis}$, $\mathcal{K}_{InProceeding}$, $\mathcal{K}_{JournalArticle}$ et $\mathcal{K}_{Article}$ sont égales à celles de l'indicateur *#concepts* de $\mathcal{K}_{Plant}$. De plus, l'indicateur *#rel-att* de Plant est égal à la somme des *#concepts* des autres contextes formels.

Structures conceptuelles incluant une association n-aire (Data11P)

Ce cas d'étude comporte un modèle relativement simple (avec 3 classes de base et une association n-aire) qui est transformé de manière « simple » en ne conservant que 3 associations sur le modèle UML_{CoBUBo} final.

Les tables 5.9 et 5.10 présentent respectivement le nombre de concepts et d'attributs relationnels pour chaque structure conceptuelle. Pour les modèles unidirectionnels et les conversions Na_{mat} et Na_{split}, le nombre total de concepts et d'attributs relationnels varie respectivement de 12 à 3 298 concepts et de 8 à 432 attributs relationnels. Pour les modèles bidirectionnels, ces nombres varient de 19 à 1 981 concepts et de 31 à 3 962 attributs. La notation « oom » signifie « manque de mémoire ». En effet, pour les modèles bidirectionnels, les treillis de concepts n'ont pu être construits à cause du manque de mémoire sur l'ordinateur. L'utilisation de modèles bidirectionnels double au moins le nombre de concepts par rapport à l'utilisation de modèles unidirectionnels. Il peut aller jusqu'à multiplier par quatre le nombre de concepts dans le cas d'AOC-posets avec Na_{split}. Généralement, l'application de Na_{split} produit plus de concepts et d'attributs relationnels que Na_{mat}, sauf pour les AOC-posets unidirectionnels.

	unidirectional models								bidirectional models							
	Na_{mat}				Na_{split}				Na_{mat}				Na_{split}			
	LAT	AOC	ICE30	ICE40	LAT	AOC	ICE30	ICE40	LAT	AOC	ICE30	ICE40	LAT	AOC	ICE30	ICE40
ProtectionSys	1340	314	4	4					oom	868	6	6				
UsedPlant	106	101	2	2	2910	233	9	7	oom	326	4	4	oom	662	46	44
TargetedOrga	206	204	3	3	344	321	5	4	oom	453	4	4	oom	892	86	84
ProtectedOrga	44	42	3	3	44	42	3	3	oom	184	5	5	oom	427	28	28
TOTAL	1696	661	12	12	3298	596	17	14	oom	1831	19	19	oom	1981	160	156

TABLE 5.9 – *Nombre de concepts (#concepts) par contexte formel produit à partir des résultats du cas d'étude Data11P, utilisant Na_{mat} et Na_{split}, pour chaque structure conceptuelle : treillis de concepts* (LAT)*, AOC-posets* (AOC) *et treillis Iceberg* (ICE30) *et* (ICE40)*. « oom » signifie « out of memory ».*

Le jeu de données correspondant concerne 398 liens de l'association ternaire (*#tri*). Avec Na_{mat}, ces liens sont représentés (matérialisés) dans le contexte $\mathcal{K}_{ProtectionSystem}$ et avec Na_{split}, ils sont principalement représentés (matérialisés) à travers le contexte $\mathcal{K}_{UsedPlant}$ et ses associations sortantes. Pour les modèles unidirectionnels, les treillis de concepts sont

	unidirectional models								bidirectional models							
	Na_{mat}				Na_{split}				Na_{mat}				Na_{split}			
	LAT	AOC	ICE30	ICE40	LAT	AOC	ICE30	ICE40	LAT	AOC	ICE30	ICE40	LAT	AOC	ICE30	ICE40
ProtectionSystem	356	347	8	8					oom	963	13	13				
UsedPlant	0	0	0	0	388	273	8	7	oom	868	6	6	oom	1319	114	112
TargetedOrganism	0	0	0	0	44	42	3	3	oom	868	6	6	oom	1089	74	72
ProtectedOrganism	0	0	0	0	0	0	0	0	oom	868	6	6	oom	1554	132	128
TOTAL	356	347	8	8	432	315	11	10	oom	3567	31	31	oom	3962	320	312

TABLE 5.10 – *Nombre d'attributs relationnels (#rel-att) par contexte formel produit à partir des résultats du cas d'étude Data11P, utilisant Na_{mat} et Na_{split}, pour chaque structure conceptuelle : treillis de concepts* (LAT)*, AOC-posets* (AOC) *et treillis Iceberg* (ICE30) *et* (ICE40)*. « oom » signifie « out of memory ».*

au moins 3 fois plus grands (1 340 pour Na_{mat} et 2 910 pour Na_{split}) et les treillis Icebergs peuvent être trop petits pour donner des informations pertinentes (4 et 9 concepts), tandis que les AOC-posets sont plus acceptables avec 314 et 233 concepts. Pour les modèles bidirectionnels, les treillis de concepts sont trop grands pour être calculés et les nombres de concepts des AOC-posets sont 1,6 à 2 fois plus grands que *#tri* pour Na_{split} et Na_{mat} respectivement. Les treillis Iceberg sont plus acceptables mais contiennent moins d'informations en raison du seuil minimal de support.

Comme illustré dans la table 5.10, si une classe du diagramme UML_{CoBUBo} est un puits, le contexte formel correspondant $\mathcal{SFC}$ ne possède aucun attribut relationnel. De plus, la classe puits correspond à la cible dans un contexte relationnel r. La valeur de *#concepts* pour $\mathcal{SFC}$ est directement ajoutée à celle de *#rel-att* du contexte formel qui est la cible de r. Par exemple, pour Na_{mat} avec le modèle unidirectionnel, les classes $\mathtt{C}_{UsedPlant}$, $\mathtt{C}_{Targeted}$ et $\mathtt{C}_{Protected}$ sont des puits. Ensuite, l'indicateur *#rel-att* du treillis $\mathcal{K}_{ProtectionSystem}$ est la somme des indicateurs *#concepts* associés aux trois autres classes. Pour les modèles bidirectionnels Na_{mat}, sans surprise, les indicateurs *#rel-att* de $\mathcal{K}_{UsedPlant}$, $\mathcal{K}_{Targeted}$ et $\mathcal{K}_{Protected}$ sont égaux à l'indicateur *#concepts* de $\mathcal{K}_{ProtectionSystem}$. De plus, l'indicateur *#rel-att* de $\mathcal{K}_{ProtectionSystem}$ est égal à la somme des indicateurs *#concepts* de $\mathcal{K}_{UsedPlant}$, $\mathcal{K}_{Targeted}$ et $\mathcal{K}_{Protected}$.

Temps de calcul et nombres d'étapes pour obtenir les structures conceptuelles

Les tables 5.11 et 5.12 présentent respectivement les nombres d'étapes et les temps d'exécution pour produire chaque structure conceptuelle en utilisant les résultats des cas d'étude DataDoc et Data11P. Pour les modèles unidirectionnels, les mêmes valeurs de *#steps* sont trouvées pour les treillis de concepts, les AOC-posets et les treillis Iceberg : 3 avec Na_{mat}, 4 avec Na_{split} et 3 pour chaque modèle concernant l'héritage. Nous observons les mêmes valeurs de *#steps* pour les représentations General, Specific et Custom (de DataDoc) : dans les modèles unidirectionnels (3) ; dans les modèles bidirectionnels et les AOC-posets (13) ; dans les modèles bidirectionnels et les treillis Iceberg (7). Pour les modèles bidirectionnels et les AOC-posets, la valeur de *#steps* de Na_{mat} (23) est environ le double de la valeur de *#steps* de Na_{split} (12). Cela ne se produit pas pour les treillis Iceberg, probablement en raison du nombre réduit de concepts. Les valeurs de *#steps* pour les modèles bidirectionnels avec les treillis Iceberg sont du même ordre de grandeur que dans les modèles unidirectionnels, tandis que pour les AOC-posets, il y a une croissance significative.

	unidirectional models					bidirectional models				
	Data11P		DataDoc			Data11P		DataDoc		
	Na_{mat}	Na_{split}	General	Specific	Custom	Na_{mat}	Na_{split}	General	Specific	Custom
LAT	3	4	3	3	3	oom	oom	oom	oom	oom
AOC	3	4	3	3	3	23	12	13	13	13
ICE30	3	4	3	3	3	5	7	7	7	7
ICE40	3	4	3	3	3	5	7	7	7	7

TABLE 5.11 – *Nombre d'étapes (#steps) pour produire les structures conceptuelles, treillis de concepts* (LAT)*, AOC-posets* (AOC) *et treillis Iceberg* (ICE30) *et* (ICE40) *avec les résultats des cas d'étude DataDoc et Data11P. « oom » signifie « out of memory ».*

		unidirectional models					bidirectional models				
		Data11P		DataDoc			Data11P		DataDoc		
		Na_{mat}	Na_{split}	General	Specific	Custom	Na_{mat}	Na_{split}	General	Specific	Custom
LAT	ADD-INTENT	0.241	0.527	13.922	12.277	12.571	oom	oom	oom	oom	oom
AOC	ARES	0.317	0.146	10.143	12.807	9.28	220	261	oom	oom	oom
	CERES	0.122	0.121	3.406	3.285	3.478	5.792	4.312	98	97	98
	HERMES	0.215	0.21	6.955	7.381	6.592	5.915	4.358	211	211	237
	PLUTON	0.168	0.137	8.432	9.056	8.444	3.587	2.216	83	94	86
ICE30	TITANIC	0.029	0.008	0.036	0.018	0.02	0.006	0.028	0.726	0.054	0.344
ICE40		0.01	0.005	0.026	0.066	0.031	0.006	0.028	0.493	0.05	0.099

TABLE 5.12 – *Temps d'exécution en secondes (#msec) pour produire les structures conceptuelles, i.e. treillis de concepts* (LAT)*, AOC-posets* (AOC) *et treillis Iceberg* (ICE30) *et* (ICE40) *avec les résultats des cas d'étude DataDoc et Data11P. « oom » signifie « out of memory ».*

Avec les modèles unidirectionnels, les mêmes valeurs de *#steps* sont trouvées pour toutes les structures (LAT, AOC, ICE), car la propagation RCA est similaire. On trouve également des valeurs similaires pour Na_{mat} et chacune des conversions de l'héritage, car les 4 modèles unidirectionnels ont la même profondeur (avec une relation allant de ProtectionSystem, pour Na_{mat} , ou Plant, pour DataDoc, vers une ou plusieurs classes).

Les temps de calcul correspondent à la somme des temps de calcul des différentes étapes. Les temps de calcul des modèles unidirectionnels sont inférieurs à 14s, ce qui est un temps raisonnable pour des experts attendant devant leur ordinateur. Mis à part pour les treillis de concepts et les AOC-posets produits avec l'algorithme ARES, les temps de calcul pour les modèles bidirectionnels sont inférieurs à 5 minutes, restant acceptables. Grâce à cette expérience, nous apprenons également quel algorithme est le plus adapté en fonction du type de jeu de données. Par exemple, si l'on veut calculer l'AOC-poset, pour les modèles unidirectionnels, CERES est le meilleur, tandis que pour les modèles bidirectionnels, PLUTON sera plus adapté. Nous observons également que, pour les modèles unidirectionnels, il n'y a pas de différence significative entre le temps de calcul pour les treillis de concepts avec ADD-INTENT et le temps de calcul pour AOC-poset avec PLUTON.

5.3.2 Évaluation qualitative

Nota : cette sous-section 5.3.2 est une traduction d'un travail paru dans l'article [Keip et al., 2019a]. Dans cette section, nous utilisons un extrait du jeu de données utilisé dans le cas d'étude Data6P pour une exploration manuelle des structures produites. Cet extrait du jeu utilisé pour Data6P s'intéresse aux mêmes 6 plantes de protection mais uniquement

contre les agresseurs de la famille des *Aspergillus*.

Pour évaluer la pertinence des transformations Na_{mat} et Na_{split}, nous avons utilisé le petit jeu de données Data6P. Nous avons ensuite exploré les AOC-posets produits par chacune des 2 transformations. Le but de cette évaluation était de répondre à une question réelle (Q1) : « Connaissant les bienfaits reconnus de *Hyptis suaveolens* dans la gestion de *Arachis hypogaea* contre *Aspergillus parasiticus*, quelles autres plantes pourraient être utilisées comme alternative ? »

La figure 5.9 donne une version simplifiée d'un extrait des AOC-posets construits à partir des contextes `BioPesticide` et `ProtectionSystem`, suite à la transformation Na_{mat}. Sur cette figure, les flèches représentent les liens de navigation (les attributs relationnels) qui ont permis de trouver et de mettre en évidence les concepts et la hiérarchie que nous souhaitons donner à notre expert du domaine pour explorer les données autour de sa question. Les numéros en gras représentent les numéros des concepts. Le texte en italique représente les objets de l'AOC-poset et le texte normal représente la cible des attributs relationnels.

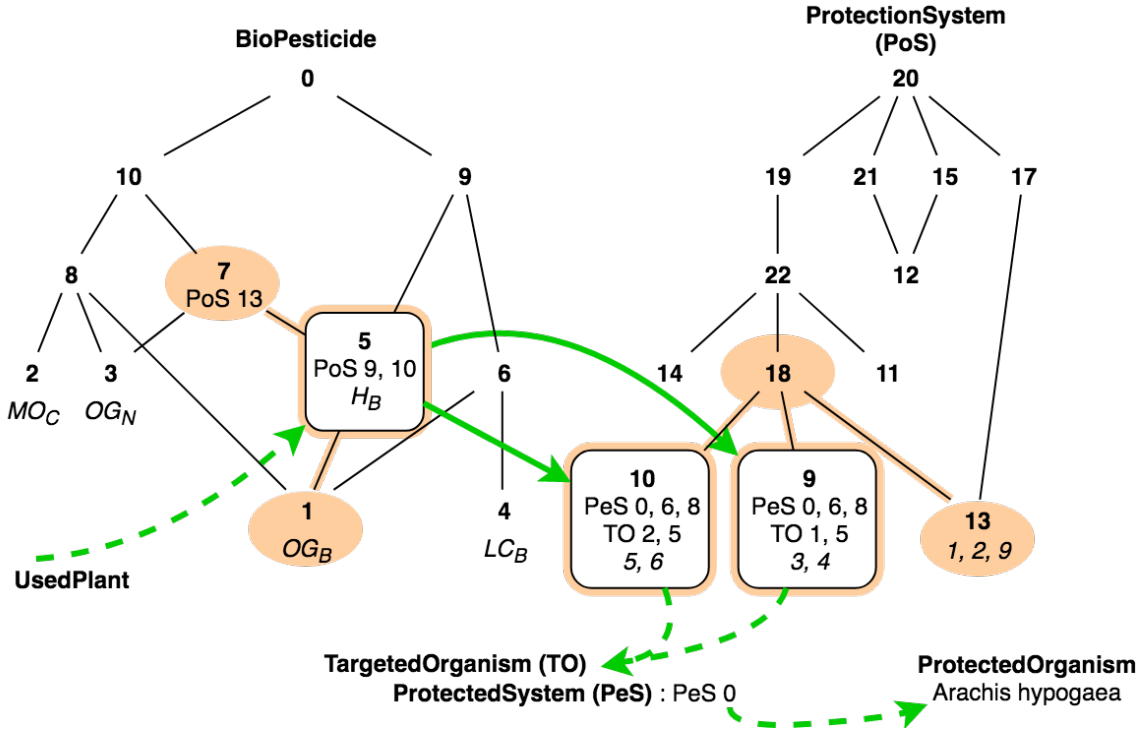


FIGURE 5.9 – *Version simplifiée d'un extrait des AOC-posets construits à partir des contextes* `BioPesticide` *et* `ProtectionSystem` *avec* Na_{mat} *pour répondre à Q1 dans l'étude de cas : les concepts parcourus et leurs liens sont mis en évidence*

En partant du concept introducteur de *Hyptis suaveolens*, de l'AOC-poset `UsedPlant` (concept non représenté), on peut aller vers les concepts 0, 10, 9, 7 et 5 de l'AOC-poset `BioPesticide` (à gauche de la Figure 5.9). Le concept le plus spécifique d'entre eux est le concept 5 qui introduit H_B, i.e. le biopesticide produit à partir de *Hyptis suaveolens* en provenance du Bénin. En suivant les attributs relationnels du concept 5, on peut ensuite aller vers les concepts 9 et 10 de l'AOC-poset `ProtectionSystem` (à droite de la Figure 5.9). Les concepts 9 et 10 sont non comparables ; leurs attributs relationnels

montrent que le concept 9 regroupe les usages de plante qui protègent *Arachis hypogaea* (`PeS 0`) de *Aspergillus ochraceus* (`TO 1`), tandis que le concept 10 regroupe les usages de plante qui protègent *Arachis hypogaea* de *Aspergillus parasiticus* (`TO 2`). Le dernier système biologique étant celui que l'on veut gérer.

Dans l'AOC-poset `BioPesticide`, nous remarquons également que le concept 5 possède un sous-concept, le concept 1, qui introduit OG_B, i.e. un biopesticide produit à partir de *Ocimum gratissimum* en provenance du Bénin. Conformément à l'ordre de spécialisation des treillis, conservé dans les AOC-posets, on peut déduire, grâce à l'hérédité des attributs, que le biopesticide produit à partir de *Ocimum gratissimum* provenant du Bénin permet de gérer au moins un des mêmes systèmes biologiques tels que présentés dans les concepts 9 et 10 de l'AOC-poset `ProtectionSystem`. *Ocimum gratissimum* peut ainsi être utilisé à la place de *Hyptis suaveolens* afin de protéger *Arachis hypogaea* contre *Aspergillus parasiticus*, mais aussi contre *Aspergillus ochraceus*, et *Aspergillus flavus*. Ces faits peuvent être vérifiés dans la base de connaissances Knomana.

De plus, le concept 5 hérite d'un attribut relationnel PoS 13 du concept 7, qui conduit au concept 13 de l'AOC-poset `ProtectionSystem`. Ce concept 13 n'est pas comparable aux concepts 9 et 10, mais ces trois concepts sont des sous-concepts du concept 18. Le concept 13 regroupe les mêmes usages de plantes que les concepts 9 et 10 (protéger *Arachis hypogaea* contre *Aspergillus flavus*), mais aussi une utilisation différente (protéger *Oryza sativa* contre *Aspergillus flavus*) en raison de l'encodage choisi. En effet, suivant le modèle obtenu avec la transformation $\mathrm{Na}_{\mathrm{mat}}$, les systèmes protégés sont d'abord classés en fonction de l'étape de production et du lieu de traitement, puis en fonction de l'organisme protégé.

Par ailleurs, le concept 1 est un sous-concept du concept 5 mais aussi d'autres concepts de `BioPesticide`. Sur la base de ces liens hiérarchiques, nous pouvons en déduire que le biopesticide produit à partir de *Ocimum gratissimum* en provenance du Bénin peut protéger d'autres systèmes biologiques que ceux précédemment décrits. L'organisation hiérarchique met en évidence ces faits pour les experts du domaine.

Considérons maintenant les AOC-posets produits avec la transformation $\mathrm{Na}_{\mathrm{split}}$. Un extrait simplifié des AOC-posets est montré sur la figure 5.10. En partant du concept introducteur de *Hyptis suaveolens* dans UsedPlant (non illustré), nous pouvons naviguer vers le concept 5 de BioPesticide (milieu de la figure 5.10) qui introduit H_B, i.e. le biopesticide produit à partir de *Hyptis suaveolens* en provenance du Bénin. Comme pour le modèle précédent, le concept 5 a un sous-concept introduisant OG_B. Les deux modèles donnent actuellement le même résultat. En allant plus loin, nous voyons que les attributs relationnels du concept 5 sont de deux types : huit d'entre eux conduisent à des concepts de `ProtectedSystem` (à droite de la Figure 5.10) tandis que les sept autres conduisent aux concepts de `Organisme Cible` (à gauche de la Figure 5.10). Les attributs conduisant aux concepts `Organisme Ciblé` révèlent, en regardant les concepts les plus spécifiques (numéro 1, 2 et 0), que les biopesticides produits à partir de *Hyptis suaveolens* et *Ocimum gratissimum* sont utilisés pour lutter contre *Aspergillus ochraceus*, *Aspergillus parasiticus* et *Aspergillus flavus*. Les attributs conduisant aux concepts `ProtectedSystem` permettent de retrouver *Arachis hypogaea* qui est introduit par le plus spécifique parmi les concepts ciblés, le concept 0, grâce à l'attribut relationnel `PeO 0`.

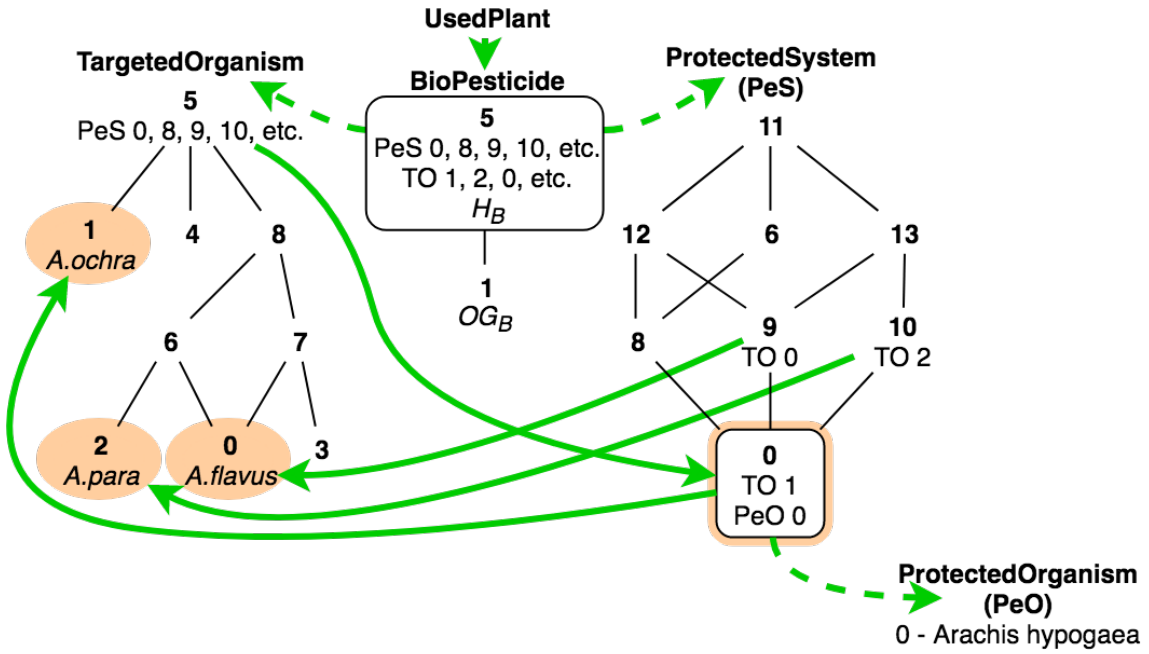


FIGURE 5.10 – *Version simplifiée d'un extrait des AOC-posets construits à partir des contextes* `TargetedOrganism` *et* `ProtectedSystem` *avec* Na_{split} *pour répondre à Q1 dans l'étude de cas : les concepts parcourus et leurs liens sont mis en évidence*

Pour résumer, pour ce cas d'étude, les deux modèles obtenus avec les transformations Na_{mat} et Na_{split} permettent de trouver *Ocimum gratissimum* comme plante alternative à *Hyptis suaveolens* pour protéger *Arachis hypogaea* contre *Aspergillus parasitique*. De plus, une requête ne donnera pas plus d'informations, contrairement à RCA qui ouvre fréquemment des hypothèses. Les concepts formés donnent non seulement une ou plusieurs réponses à la question initiale, mais ils montrent également comment ces réponses sont classées et quelle description supplémentaire (non incluse dans la question initiale) ils partagent. En effet, les deux modèles montrent également que *Hyptis suaveolens* peut être remplacé par *Ocimum gratissimum* pour protéger *Arachis hypogaea* contre *Aspergillus ochraceus* et *Aspergillus flavus*, qui est une information complémentaire. Par ailleurs, en examinant le voisinage des concepts qui donnent la réponse recherchée, les experts peuvent ainsi formuler des hypothèses pour de nouvelles recherches. Par exemple, s'ils remarquent qu'une plante protège un organisme ciblé contre un agresseur spécifique, ils peuvent concevoir des expériences pour évaluer si d'autres plantes ayant des caractéristiques similaires (regroupées dans le même concept) peuvent également avoir le même effet. Cependant, un ensemble de trois relations binaires n'est pas équivalent à une relation ternaire. Le modèle utilisant Na_{split} sera donc parfois moins précis car il manque cette relation ternaire. De plus, la navigation y est plus difficile en raison du plus grand nombre de concepts et d'attributs relationnels.

5.3.3 Modélisation des données indéterminées par FCA et ses extensions

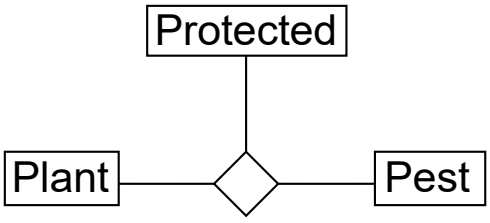
Dans les sections précédentes, l'évaluation des conversions a porté sur les RCFs produites pour RCA. Nous avons également comparé des conversions d'associations n-aires pour RCA avec deux extensions de FCA, i.e. TCA et G-FCA présentées dans le chapitre 2. Pour ce faire, le cas d'étude utilisé est DataSpodo, qui, en plus de l'association n-aire dans le modèle de données, comporte des données indéterminées. Ce cas d'étude nous permet donc de vérifier la pertinence de notre méthode par rapport à la conversion des données indéterminées.

Cas d'étude (DataSpodo) Le jeu de données pour ce cas d'étude concerne 3 espèces du genre *Spodoptera* : *S. littoralis*, *S. litura*, et *Spodoptera* spp. Il rassemble 12 descriptions d'usages de plante concernant 6 organismes protégés et 9 plantes de protection. Le principe de ce cas d'étude est le suivant. TCA est l'extension relationnelle de FCA la plus à même de conserver les propriétés d'une association ternaire car cette extension permet justement de mettre en relation trois groupes d'objets. Nous allons donc analyser certains concepts obtenus avec TCA pour vérifier la cohérence de la conversion des données indéterminées. Puis, en partant de ces concepts triadiques, nous verrons si nous sommes capables de retrouver les notions contenues dans ces concepts dans les structures conceptuelles créées par les deux autres méthodes (RCA et GFCA). Si tel est le cas, cela contribue à conforter l'idée que la conversion des données indéterminées et la conversion de l'association ternaire sont pertinentes.

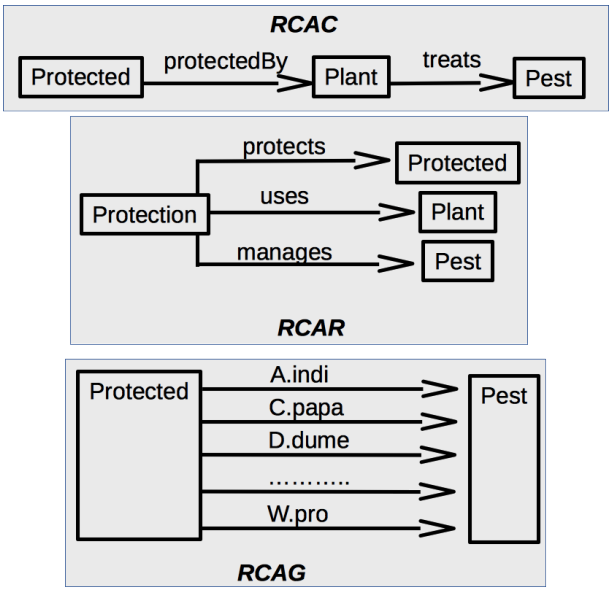
La figure 5.11 présente tous les modèles de représentation d'une association ternaire utilisés pour cette évaluation. Le modèle illustré par la figure 5.11a représente le modèle original. Il met en relation les trois éléments présents, i.e. une plante de protection ($\mathtt{C}_{Plant}$), un système protégé ($\mathtt{C}_{Protected}$) et un agresseur ($\mathtt{C}_{Pest}$), en utilisant une association ternaire. Il s'agit également du modèle utilisé pour la méthode TCA, chaque classe étant convertie en un des 3 ensembles d'objets de TCA. La figure 5.11b illustre les 3 autres modèles utilisés. Chacun des 3 modèles a été utilisé pour la méthode RCA. Le premier (RCAC) utilise une chaîne entre les 3 éléments. Les organismes protégés sont ainsi classés en fonction des plantes utilisées pour les protéger, elles-mêmes classées en fonction des agresseurs qu'elles traitent. Le deuxième (RCAR) utilise l'opération $\mathrm{Na}_{\mathrm{mat}}$. Le troisième (RCAG) instancie une relation binaire entre deux des catégories d'objets par les objets de la troisième catégorie. Ce dernier modèle est d'ailleurs celui utilisé pour GFCA. Pour les modèles RCAC et RCAG il est possible d'interchanger les éléments en fonction de ce que l'on souhaite obtenir, par exemple pour répondre à une question particulière.

La table 5.13 montre l'intégralité du contexte triadique utilisé pour TCA après conversion des données indéterminées. TCA étant ici utilisé pour représenter l'association ternaire, il n'est pas possible d'ajouter des attributs à la catégorie Pests. Le *scaling* a donc été effectué dans les objets de la catégorie Pests. Les cases rouges représentent le concept triadique **TC5** présenté dans la suite.

TCA est une extension de FCA qui a été spécialement conçue pour faire le lien entre 3 ensembles. On peut donc considérer qu'elle est la méthode la plus fidèle à la relation



(a) *Modèle de données original. Utilisé pour la méthode TCA.*



(b) *Modèles utilisés pour la méthode RCA. Le modèle RCAG est également utilisé pour la méthode G-FCA.*

FIGURE 5.11 – *Modèles utilisés pour le cas d'étude DataSpodo. (a) Le modèle original, également utilisé pour TCA. (b) Les modèles utilisés pour RCA : une chaîne (RCAR), Na_{mat} (RCAR) et un ensemble d'associations binaires instanciées par un des 3 éléments (RCAG). RCAG est également utilisé pour G-FCA.*

S. litu	W. pros	V. cane	V. fusc	V. parv	A. indi	C. spp	C. opul	C. papa	D. dume
Z. mays									
A. escu									
B. ole									
G. hirs									
S. lyco									
R. com	×								

S. spp	W. pros	V. cane	V. fusc	V. parv	A. indi	C. spp	C. opul	C. papa	D. dume
Z. mays					×	×	×		
A. escu									
B. ole									
G. hirs									
S. lyco									
R. com									

S. litto	W. pros	V. cane	V. fusc	V. parv	A. indi	C. spp	C. opul	C. papa	D. dume
Z. mays									
A. escu					×			×	
B. ole					×			×	
G. hirs									×
S. lyco		×	×	×					
R. com									

S.litu+spp	W. pros	V. cane	V. fusc	V. parv	A. indi	C. spp	C. opul	C. papa	D. dume
Z. mays					×	×	×		
A. escu									
B. oler									
G. hirs									
S. lyco									
R. com	×								

S.spp+any	W. pros	V. cane	V. fusc	V. parv	A. indi	C. spp	C. opul	C. papa	D. dume
Z. mays					×	×	×		
A. escu					×			×	
B. oler					×			×	
G. hirsu									×
S. lyco		×	×	×					
R. com	×								

S.litto+spp	W. prost	V. cane	V. fusc	V. parv	A. indi	C. spp	C. opul	C. papa	D. dume
Z. mays					×	×	×		
A. escu					×			×	
B. oler					×			×	
G. hirs									×
S. lyco		×	×	×					
R. com									

TABLE 5.13 – *Contexte triadique obtenu après scaling des données indéterminées dans la liste des objets de la catégorie Pests. Les cases rouges représentent le concept triadique* ***TC5*** *présenté dans la suite.*

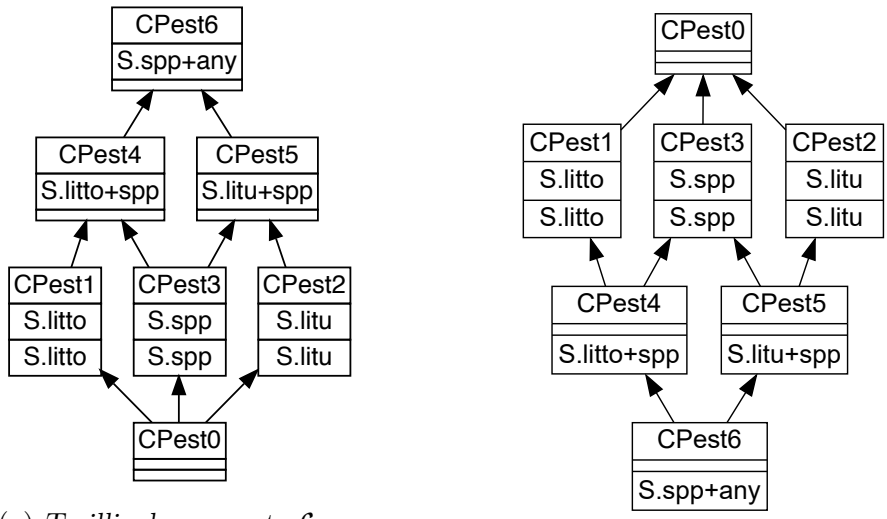
ternaire parmi les 3 extensions choisies. Pour cette évaluation, 3 concepts triadiques (des concepts de la méthode TCA) ont donc été sélectionnés. Nous avons choisi ces concepts car ils ne sont pas triviaux et fournissent des informations de valeur sur les données. L'objectif de l'évaluation était, en premier lieu, de voir si ces 3 concepts sont présents dans les résultats des deux autres méthodes. Dans un deuxième temps, cela permet de comparer les différentes caractéristiques de chacune des 3 méthodes. La table 5.14 présente les trois concepts sélectionnés. **TC2** indique qu'aucune plante ne protège tous les organismes contre toutes les espèces de *Spodoptera* du jeu de données. **TC5** groupe A.escu et B.ole, qui sont protégés contre S.litto par A.indi et C.papa. Cela montre, par exemple, que A.indi et C.papa peuvent se remplacer l'un l'autre pour contrôler S.litto. Enfin, le dernier concept, **TC6**, nous montre que A.indi contrôle certaines espèces de *Spodoptera* dont potentiellement S.litto, sur les trois plantes A.escu, B.ole et Z.mays. Ce concept **TC6**

n'existe que grâce à la conversion des données indéterminées comme le montre la liste des Pests du concept. Cette conversion a donc permis de produire un nouveau concept ayant du sens et ce concept pourra orienter les experts vers de nouvelles solutions.

TC2	**Protected Plants**	A.escu B.ole G.hirsu R.com S.lyco Z.mays
	Pests	S.litto S.litto+spp S.litu S.litu+spp S.spp S.spp-any
TC5	**Protected Plants**	A.escu B.ole
	Pests	A.indi C.papa
		S.litto S.litto+spp S.spp-any
TC6	**Protected Plants**	A.escu B.ole Z.mays
	Pests	A.indi
		S.litto+spp S.spp-any

TABLE 5.14 – *Trois concepts triadics sélectionnés parmi les 9 construits à partir du contexte triadic de la table 5.13*

Conversion des données indéterminées pour RCA Nous avons vu dans la sous-section 4.4.3 que le *scaling* des données indéterminées pour RCA pouvait être appliqué sur l'ensemble des objets ou des attributs. La figure 5.12 montre les treillis obtenus avec l'une et l'autre des deux versions du *scaling*. Dans ce cas d'étude, c'est la version de la sous-figure 5.12a qui a été appliquée.



(a) *Treillis de concepts $\mathcal{L}_{Pest}$ après scaling dans les attributs*

(b) *Treillis de concepts $\mathcal{L}_{Pest}$ après scaling dans les objets*

FIGURE 5.12 – *Treillis de concepts $\mathcal{L}_{Pest}$ après scaling dans les attributs (à gauche) et les objets (à droite).*

Résultat de la comparaison des trois méthodes Au final, sur les 3 concepts sélectionnés, 2 sont présents dans toutes les classifications. Les concepts TC5 et TC6 sont présents dans toutes les classifications, mais pas le concept TC2. Il n'apparaît pas dans RCA car tous les organismes sont protégés par au moins une plante à cause de l'utilisation du quantifieur existentiel. En utilisant un autre quantifieur, il serait éventuellement possible de retrouver TC2 dans les concepts de RCA.

La table 5.15 représente le comparatif entre les 3 méthodes. Les cases vertes indiquent que l'élément est permis par la méthode, les rouges que l'élément n'est pas possible et les bleues qu'il l'est sous certaines conditions. Pour cette comparaison, nous avons observé plusieurs critères. Le premier concerne la possibilité d'afficher la totalité des résultats de la méthode sur une seule structure. Sur ce point, RCA est la seule méthode qui ne le permette pas. Inversement, RCA est la seule méthode à proposer une classification par catégorie. Tout dépend donc du point de vue qui intéresse l'expert à un moment donné de son analyse. Vient ensuite la possibilité de représenter nativement des associations n-aires avec la méthode. Là encore, RCA est le seul qui doive absolument utiliser des associations binaires. Cependant, on peut représenter des relations n-aires avec RCA en utilisant des méthodes telles que la matérialisation. De son côté, TCA ne permet pas de représenter des associations n-aires, mais il existe une autre version apte à cela, i.e. Polyadic Concept Analysis ([Voutsadakis, 2002]). TCA ne permet pas non plus l'ajout d'attributs sur les objets à partir du moment où les 3 ensembles servent chacun à représenter une catégorie d'objets, et cette méthode ne permet pas non plus d'avoir plusieurs relations entre des objets ni de varier les représentations comme nous avons pu le faire avec RCA. La récupération des données originales est directe avec TCA mais demande l'utilisation d'attributs spécifiques pour RCA et GFCA. Enfin RCA est la seule méthode qui permette la génération de nouvelles hypothèses grâce à l'utilisation des quantifieurs.

Criteria	**TCA**	**RCA / ∃**	**GFCA**
Structure	1 lattice	n lattices	1 lattice
Classification per category			
Ternary implementation			
Extension to N-ary	Polyadic		
Attributes associated to an object	applied to 3 object sets		
Several relations between objects			
Various encodings			
Can return to original data		with id attributes	with id attributes
Generate hypotheses			
Logical operators			

TABLE 5.15 – *Comparatif entre trois extensions relationnelles de FCA sur le cas d'étude DataSpodo. Les cases vertes indiquent que le critère est satisfait par la méthode, les rouges qu'il ne l'est pas et les bleues qu'il l'est sous certaines conditions.*

Pour résumer, TCA est donc plus restrictif que RCA et GFCA mais permet de revenir aux données d'origine. L'utilisation du quantifieur existentiel rend RCA moins précis que les autres extensions (sauf si tous les objets ont un identifiant, dans ce cas on peut revenir aux données d'origine). Mais il permet aux experts d'accéder à des hypothèses et de proposer de nouvelles solutions de protection.

5.4 Évaluation de la validité de l'approche expérimentale

Dans cette section, nous évaluons la validité de l'approche expérimentale. Premièrement, nous étudions la validité interne, en examinant la façon dont nos données sont collectées et en considérant la fiabilité de nos outils afin d'identifier les préoccupations et les biais possibles. Deuxièmement, nous analysons la validité de construction. À cette fin, nous examinons la pertinence des mesures que nous utilisons pour évaluer les conclusions. Troisièmement, nous considérons la validité des conclusions afin de vérifier que les conclusions que nous tirons de nos résultats sont raisonnables. Ces conclusions doivent exprimer une relation causale entre les observations et les conclusions. Enfin, nous étudions la validité externe, c'est-à-dire si les résultats peuvent être généralisés à d'autres situations.

Validité interne Les données de Knomana sont collectées et contrôlées manuellement par deux experts du domaine qui font partie de l'équipe de recherche. Pour contrôler la validité des programmes d'extraction, qui produisent des fichiers conformes au format d'entrée RCAexplore, une vérification manuelle a été effectuée sur chacun des jeux de données. Dans le cas de Data11P et DataDoc, une légère divergence a été remarquée pendant la vérification des résultats. Elle est due à l'absence d'une tâche de pré-traitement (sur le fichier CSV) pour homogénéiser les valeurs similaires (accents, espace, etc.) et au traitement de certaines valeurs présentes dans le fichier original, e.g. #N/A.

La production des structures conceptuelles a été réalisée grâce aux algorithmes inclus dans RCAexplore. Cet outil a été utilisé dans des expériences antérieures avec des résultats validés. En particulier, le calcul des AOC-posets est effectué à travers plusieurs algorithmes dont les résultats ont été comparés pour vérifier qu'ils étaient identiques. Les métriques sont calculées à partir des sorties de RCAexplore et certaines ont été vérifiées manuellement.

Validité de construction Les métriques utilisées pour évaluer l'applicabilité nous permettent de vérifier que les transformations sont applicables sur nos données. Pour ce faire, elles comptent le nombre de cas d'application, non seulement dans le modèle, mais aussi sur les instances.

Les métriques de mesure du passage à l'échelle évaluent la taille des structures conceptuelles et le temps de calcul, qui sont deux aspects principaux pour les experts. La taille nous permet d'évaluer si l'utilisation de la structure conceptuelle à des fins d'analyse nécessite un outil d'analyse supplémentaire. Les temps de calcul alertent les experts sur la faisabilité et les conditions d'utilisation. Par exemple, ils savent si le temps de calcul est proche de quelques minutes ou de quelques heures, ou si le calcul n'a aucune chance de se terminer.

La taille des structures conceptuelles et le temps de calcul aident également un expert à choisir la structure la plus pertinente : e.g. les treillis Icebergs sont pertinents pour les

motifs fréquents avec de petites structures et un temps de calcul réduit.

Validité des conclusions Les conclusions sur l'applicabilité sont raisonnables concernant la relation ternaire. Les relations n-aires sont assez rares dans les modèles, mais si elles sont centrales, comme dans Knomana, elles concernent de nombreuses instances.

Une première conclusion pour le passage à l'échelle concerne toutes les conversions. Cette expérimentation montre les limites que peut connaître l'évolution de notre approche, sur ordinateur portable, sur certaines configurations de données. Nous avons déjà observé que, sur les modèles bidirectionnels, les treillis de concepts sont à peine calculés et ils sont même impossibles à calculer dans d'autres cas ([Keip et al., 2019a, Miralles et al., 2015]). Deuxièmement, les remarques faites sur les correspondances entre les indicateurs *#concepts* et *#rel-att* pour les classes puits peuvent être fondées sur la définition mathématique de RCA.

Validité externe Nous considérons la validité externe sur plusieurs aspects : représentativité du jeu de données (données et modèle de données), étendue des transformations, généralité des observations.

Nous avons appliqué notre approche sur plusieurs cas d'études qui sont des extraits représentatifs de Knomana. Knomana est lui-même représentatif de connaissances provenant de publications scientifiques dans le domaine de l'environnement : organismes vivants et leurs caractéristiques, documents décrits avec des relations incluant la spécialisation (is-a). DataDoc est typique des bases de données bibliographiques telles que Zotero ou BibTeX. Les extraits ne sont pas très volumineux. Mais dans de nombreuses analyses qualitatives, les experts considèrent des jeux de données restreints, ce qui justifie de considérer un tel volume de données.

Les cas d'étude possédant une relation ternaire (e.g. Data11P), exposent une relation n-aire, qui est une construction typique de la modélisation dans le paradigme orienté objet et aussi dans le paradigme des graphes de connaissances, tels que les graphes conceptuels, et les ontologies avant la grande diffusion de RDF. En conséquence, ce type de modèle est présent dans de nombreux jeux de connaissances existants et permet d'exprimer des données avec toute leur complexité intrinsèque. Par ailleurs, comme en RDF, notre modèle de données DataDoc intègre des hiérarchies de spécialisation (is-a).

Les transformations proposées ne sont spécifiques ni à Knomana, ni à RCA. Dans le processus de construction d'un modèle conforme à l'entrée RCA, nous modifions le modèle de données avec des opérations qui seraient les mêmes pour transformer les modèles et les données en schéma binaire (comme RDF).

Les observations sur l'effet des transformations sur le passage à l'échelle seraient les mêmes pour les méthodes de découverte de connaissances d'ingénierie du même type, par exemple le calcul de bases d'implication ou l'extraction de règles d'association [Bertet et al., 2018].

5.5 Conclusion

Ce dernier chapitre a présenté différentes évaluations des conversions sur des jeux de données réels. Guidés par des critères et indicateurs (section 5.1), nous avons effectué des évaluations quantitatives sur les résultats des conversions (section 5.2). En section 5.3, nous avons effectué des évaluations quantitatives et qualitatives sur les résultats des conversions mais également sur les structures conceptuelles produites par ces résultats afin d'en attester la pertinence. Enfin nous avons étudié la validité de notre approche expérimentale (section 5.4). Outre le fait de donner des éléments de validation des conversions, les évaluations qualitatives sont aussi une contribution au domaine de la découverte de connaissances, montrant l'intérêt des méthodes basées sur l'Analyse de Concepts Formels sur des données environnementales.

VI

Conclusion et perspectives

Ce mémoire a présenté les principaux travaux effectués dans le cadre de cette thèse. Ce dernier chapitre propose un résumé des contributions apportées, en section 6.1, puis, en section 6.2, discute de questions ouvertes et de travaux futurs.

6.1 Résumé des contributions

Les travaux effectués s'inscrivent dans le contexte du développement de systèmes à base de connaissances et ont été appliqués au système à base de connaissances Knomana. Knomana regroupe plusieurs jeux de connaissances, dont PPAf qui rassemble des connaissances sur les usages de plantes ayant un effet pesticide ou antibiotique, pour protéger l'être humain (santé humaine et santé publique), l'élevage (santé animale) ou une culture agricole (santé végétale). Le modèle de données de PPAf a servi de support de réflexion pour élaborer les méthodes et ses données ont été utilisées pour évaluer ces méthodes.

Les méthodes de fouille de données privilégiées pour l'exploration de cette base sont l'Analyse de Concepts Formels et ses extensions. Ces méthodes ont été choisies pour leur aptitude à inclure différents aspects de l'exploration de jeux de données en lien avec les sciences du vivant, à l'exemple des relations entre objets, de la temporalité ou du flou. De plus, l'Analyse de Concepts Formels permet également d'extraire les connaissances sous forme de règles d'implications qui s'apparentent à celles utilisées par les moteurs d'inférence pour raisonner sur les connaissances dans un objectif d'aide à la décision.

La transformation du jeu de données dans le format d'entrée attendu est une étape indispensable du processus de découverte de connaissances et qui est cruciale pour la qualité des résultats. Dans le cas de PPAf, nous avons ainsi observé l'indétermination de certaines données et la présence de relations complexes (n-aires) dans le modèle de données. L'objectif du travail n'était pas de se restreindre à ces problématiques de PPAf, mais de proposer des solutions qui soient réutilisables dans d'autres jeux de connaissances

de Knomana, voire dans d'autres bases de connaissances.

La première problématique traitée a concerné la conversion automatique d'un jeu de connaissances, dont le modèle de données est formalisé avec le langage UML, sous la forme d'une famille de contextes relationnels. La seconde problématique a concerné les diverses modélisations possibles d'une relation n-aire et leur impact sur la classification des connaissances. La dernière problématique a concerné la levée de l'indétermination des données.

Un catalogue d'opérations de conversion de modèle dans un cadre unifié Nous avons proposé un catalogue d'opérations de conversion. Certaines sont des opérations ou des variantes d'opérations que l'on peut rencontrer dans la littérature en bases de données, en génie logiciel ou en représentation des connaissances. Nous les avons décrites dans un cadre formel unifié. De façon à permettre une compréhension plus abordable des descriptions formelles accompagnant chaque opération, nous donnons, pour chaque opération, une description de son comportement. Cette description est illustrée en UML avec un diagramme de classes succinct et un diagramme d'objets représentatifs de l'élément à convertir. Ce n'est qu'ensuite que la description formelle est présentée.
De plus, des exemples sont également proposés afin de permettre une meilleure compréhension des effets des opérations de conversion qui vont d'un diagramme de classes UML vers une famille de contextes relationnels.

Algorithme général pour le processus de conversion Le processus de conversion est mis en œuvre dans un algorithme décrivant toutes les étapes et ordonnant l'application des opérations. Cet algorithme est composé de trois étapes : (1) transformation de l'héritage puis des associations, (2) *scaling* des attributs multi-valués et (3) conversion en famille de contextes relationnels (RCF). Nous l'avons vu précédemment, les méthodes de la littérature n'accompagnent pas toujours les opérations de conversion de recommandation ni d'algorithme général de mise en œuvre comme nous l'avons fait ici.

Conversion de l'héritage En s'appuyant sur l'expérience des travaux en génie logiciel et en science des données, notre méthode permet la conversion de l'héritage pour les langages dans lesquels cette notion n'est pas native. Pour ce faire, la hiérarchie est aplatie à un niveau de la hiérarchie choisi par l'expert. Pour ne pas perdre les caractéristiques spécifiques aux sous-classes du niveau choisi, ces caractéristiques sont remontées dans leur super-classe. Si des auteurs de la littérature proposent déjà des opérations consistant à aplatir l'héritage ou à fusionner des classes directement connectées par héritage, nous n'avons pas trouvé de solution qui soit paramétrable à un niveau quelconque de la hiérarchie tel nous le proposons.

Conversion d'une association n-aire Nous avons réutilisé et approfondi des solutions de conversion pour les associations n-aires. Ces solutions incluent la matérialisation, qui remplace l'association par une nouvelle entité (une nouvelle classe en UML) puis qui relie cette entité aux extrémités de l'association n-aire.

Nous présentons également des conversions d'associations ternaires, dont le résultat produit un modèle avec une sémantique plus faible (i.e. des informations sont perdues) pour privilégier la génération d'hypothèses pour les experts lors de l'application de la méthode de fouille. La première conversion décompose l'association n-aire pour produire un graphe complet de relations binaires entre les entités. La deuxième produit une représentation en chaîne où chaque entité est reliée à la suivante par une association binaire (e.g. l'organisme protégé est protégé par une plante qui traite un agresseur). La troisième produit un ensemble de relations binaires entre deux organismes (e.g. culture et agresseur) et chacune de ces relations est étiquetée avec un objet du troisième organisme (plante). En fonction de l'élément que l'on veut faire ressortir, on peut modifier la configuration de la transformation, ce qui a pour effet de produire des classifications différentes.

Autres éléments de conversion des associations En plus de l'héritage et des associations n-aires, des conversions concernant les autres types d'associations sont présentées. La conversion d'une classe d'association utilise le même principe que la matérialisation d'une association n-aire. Les agrégations, compositions et dépendances sont converties en associations simples. Enfin, une association bidirectionnelle est séparée en deux associations unidirectionnelles de directions inverses.

Un nouveau type de *scaling* Pour l'Analyse de Concepts Formels, le *scaling*, qui transforme les attributs multi-valués en ensembles d'attributs booléens est un point essentiel de modélisation. Il existe deux types de *scaling* principaux : le *scaling* nominal où chaque valeur de l'attribut multi-valué est mise en correspondance avec un seul attribut booléen, et le *scaling* ordinal, où chaque valeur de l'attribut multi-valué est mise en correspondance avec sa description (e.g. 2010 étant plus grand 2000, la valeur 2010 est décrite par ≥ 2000 et ≥ 2010).
À partir de ces deux types de *scaling* principaux, nous avons défini notre propre *scaling*, le *scaling* par intervalles. Cette version utilise des intervalles de valeurs bornés qui peuvent se chevaucher. Par exemple, l'attribut 2002 va être décrit par les intervalles [2000,2010] et [2000,2005].

Modélisation des données indéterminées Dans le monde du vivant, la nomenclature scientifique utilise des abréviations pour désigner des espèces indéterminées ou un groupe d'espèces du même genre, i.e. « sp. » et « spp. ». Les abréviations sp. et spp. représentent respectivement une espèce indéterminée d'un genre, et au moins deux espèces d'un genre. Lorsqu'une connaissance comporte une telle donnée indéterminée, cela entraîne de l'indétermination sur la connaissance. Pour modéliser les données indéterminées utilisant l'abréviation « spp. », nous avons défini une hiérarchie à trois niveaux d'agrégation. Au bas de la hiérarchie se trouvent les noms d'espèce du même genre présents dans la base PPAf. Au-dessus, chaque nom d'espèce est combiné avec l'espèce indéterminée. Ce niveau attribue également à chacune des espèces les informations concernant spp. Enfin, le niveau le plus haut regroupe toutes les données du genre concerné. On obtient ainsi le niveau d'agrégation le plus général. Pour utiliser cette hiérarchie dans notre méthode de conversion, nous l'appliquons au même titre qu'une opération de *scaling*.

Évaluations quantitatives et qualitatives de la méthode sur des cas d'étude représentatifs Pour évaluer nos propositions, nous avons appliqué notre approche sur plusieurs cas d'étude correspondant à des extraits représentatifs du contenu de Knomana. La première série d'évaluations concerne la conversion du modèle UML en famille de contextes relationnels (RCF). Ces évaluations sont menées sur quatre cas d'étude. Nous avons notamment évalué l'impact des conversions sur la taille des familles de contextes relationnels produites en observant les nombres d'objets et d'attributs booléens obtenus. Le premier cas d'étude concerne la conversion de l'héritage appliquée au modèle de description des documents insérés dans Knomana. Les deuxième et troisième cas d'étude concernent la conversion de la relation n-aire. Le quatrième cas d'étude concerne le jeu de données de PPAf dans son intégralité et la conversion d'une classe d'association. La deuxième série d'évaluations a été menée sur les structures conceptuelles produites sur les familles de contextes relationnels obtenues suite aux conversions. Nous avons en particulier mesuré l'effet des conversions sur les structures produites en observant les nombres de concepts et d'attributs relationnels produits par RCA ainsi que le nombre d'étapes et le temps d'exécution nécessaires à RCA pour produire les structures. La dernière évaluation a porté sur la comparaison de trois extensions relationnelles de FCA en considérant leur capacité de traitement d'une relation ternaire et la modélisation des données indéterminées.

Méthode générique Les différents types de modèle utilisés (avec héritage ou relation n-aire) sont présents dans de nombreux jeux de connaissances existants et permettent d'exprimer des données avec toute leur complexité intrinsèque. Par conséquent, les transformations proposées ne sont spécifiques ni à Knomana, ni à l'Analyse de Concepts Formels ou Relationnels (FCA/RCA), car elles peuvent être appliquées à d'autres langages. En effet, dans le processus de construction d'un modèle conforme à l'entrée de FCA/RCA, nous convertissons le modèle de données avec des opérations qui seraient proches ou identiques à celles utilisées pour transformer les modèles et les données vers un schéma binaire pour des processus de fouille comme la construction de règles d'association ou des arbres de décision.

Développements logiciels Le travail conduit au cours de la thèse a mené au développement de différents outils. Ils ont été assurés par Alain Gutierrez, ingénieur d'études au CNRS, par des stagiaires de master 1 et 2, ou par moi-même. Ainsi, pour permettre une extraction plus simple des données de Knomana (organisées dans un tableur), un premier prototype logiciel paramétrable à l'aide d'un fichier (de format .json) a été développé. Le fichier contient : le nom du fichier Excel d'entrée, le nom du fichier de sortie (format .rcft), et la structure des contextes formels et relationnels souhaités. La description d'un contexte formel requiert son nom, les numéros des colonnes créant les objets et des colonnes comportant les attributs. À partir des colonnes sélectionnées, un *scaling* nominal permet de générer les attributs booléens. La description d'un contexte relationnel nécessite le nom de la relation, le quantifieur (pouvant être modifié ultérieurement), le nom des contextes formels source et cible, et si nécessaire, le nom de la relation inverse. Indiquer le nom de la relation inverse entraîne la création automatique du contexte relationnel inverse. Ce prototype a ensuite été repris et amélioré par Alain Gutierrez, puis intégré dans

la plateforme CoGui [1]. Les algorithmes en ont ensuite été extraits et rendus disponibles pour faciliter la réplication des expériences dans la bibliothèque FCA4J [2].

Concernant la méthode de conversion présentée dans ce mémoire, une première partie de prototype logiciel a été développée par Bashar Rima et Joseph Saba à l'occasion d'un projet de Master Informatique. Ce prototype [3] met en œuvre notre méthode via une architecture générique de transformation de modèle.

Production de données Ce travail, réalisé en équipe, a mené au dépôt de plusieurs jeux de données à l'occasion de soumissions d'articles. Ces données sont disponibles sur le dataverse du CIRAD (ils sont cités à la fin de l'introduction). L'un d'eux représente un extrait de PPAf sur les espèces de *Noctuidae* ayant servi pour la génération de règles d'implication. Un second contient la base PPAf anonymisée ayant servi à l'évaluation de la conversion sur un large volume de données et en utilisant la conversion d'une classe d'association. Le dernier contient un extrait de PPAf sur les espèces de *Spodoptera* ayant servi pour la comparaison des trois extensions relationnelles de FCA.

6.2 Perspectives

Ce travail a suscité de nombreuses questions et nous n'avons pas pu toutes les traiter. Nous évoquons les principales ci-après.

Données indéterminées La méthode de levée de l'indétermination des données proposée semble pertinente en regard de l'interprétation des résultats. Nous avons proposé deux modélisations pour les données indéterminées. L'une est appliquée comme un *scaling* et crée de nouveaux attributs. Il s'agit de celle appliquée pour la comparaison des extensions relationnelles de FCA. L'autre applique le *scaling* dans la liste des objets du contexte, il reste encore à l'évaluer. Il serait intéressant de comparer l'impact de ces deux modélisations sur un même jeu de données et surtout sur un jeu de données de taille plus importante. Cela permettrait d'évaluer leur équivalence ou leur divergence sur les classifications obtenues et de déterminer les situations où l'une serait mieux adaptée que l'autre.

Multiplicité des associations n-aires La multiplicité est un élément des associations que nous n'avons pas spécifiquement pris en compte dans notre méthode. Pourtant, elle pourrait avoir un effet sur la conversion, et donc sur les diverses conversions possibles d'une relation n-aire. Aussi, considérer cet aspect permettrait d'offrir un cadre d'application plus général de la méthode de conversion, basée sur les décompositions élémentaires à l'exemple de celles mentionnées dans l'état de l'art avec l'article [Jones and Song, 2000]. En effet, l'étude des relations logiques implicites de la relation ternaire permet aux auteurs

1. http://www.lirmm.fr/cogui/
2. https://www.lirmm.fr/fca4j/
3. https://github.com/anonbnr/UML2RCA

d'identifier les décompositions binaires équivalentes possibles. Ces décompositions sont donc des alternatives à la réification (matérialisation) qui permettent de conserver la sémantique de la relation n-aire.

Nommer les relations La prise en compte des identifiants figurant sur les associations du diagramme de classes UML, i.e. le nom de l'association ou le rôle des extrémités, permettrait de nommer automatiquement les associations créées par les opérations de conversion, et en particulier lors de la conversion d'une association bidirectionnelle. Par exemple, imaginons qu'une association bidirectionnelle entre deux classes Plante et Agresseur soit nommée « traite ▶ ». De façon automatique, ce nom pourrait être utilisé pour nommer l'association allant de Plante vers Agresseur avec le nom « traite », tandis que l'association inverse serait nommée « traitéPar » (un dictionnaire ou une ressource lexicale comme JeuxDeMots [4] pouvant être mobilisés pour trouver ce nom inverse), ou encore par un nom générique construit comme « traite^{-1} ».

Évaluation de la pertinence de la conversion de l'héritage sur le contenu des structures conceptuelles Dans ce mémoire, nous proposons une méthode de formulation de l'héritage dans une famille de contextes relationnels, via la propagation des attributs et des objets de la hiérarchie. L'évaluation qualitative des structures conceptuelles produites à partir de cette famille de contextes relationnels est à conduire pour vérifier la pertinence de cette conversion au niveau de la classification, en s'assurant que l'on sera capable, dans les structures construites, de retrouver la notion d'héritage.

Évaluation des transformations dans le cas d'une association réflexive Dans la conversion de l'héritage, les sous-classes d'une classe choisie comme niveau de représentation sont fusionnées dans celle-ci. Dans le cas où une de ces sous-classes comporterait une association avec sa super-classe, la fusion entraînera la création d'une association réflexive.
En première approche de la conversion, nous n'avons pas détecté de problème associé à la présence d'une association réflexive dans le modèle de données. En effet, si une classe (abstraite ou concrète) possédant une association réflexive est la cible d'une transformation, alors la classe résultante de la transformation possédera également cette association réflexive. Une évaluation qualitative sur les structures conceptuelles permettrait de vérifier le comportement d'une telle association dans la famille de contextes relationnels.

Développement d'un prototype pour les experts L'objectif général du travail est de développer une structure de système à bases de connaissances. Comme le montre le diagramme de composants de la section 1.3 reporté ici en figure 6.1, l'architecture envisagée est composée de deux parties : la base de connaissances et le moissonneur conceptuel. Le travail présenté dans ce mémoire concerne l'alimentation du composant Reasoning Tool, qui a pour fonction de compiler les connaissances et les structurer. Autrement dit, son rôle consiste à préparer le jeu de connaissances, résultant ou non de la combinaison de

4. http://www.jeuxdemots.org/

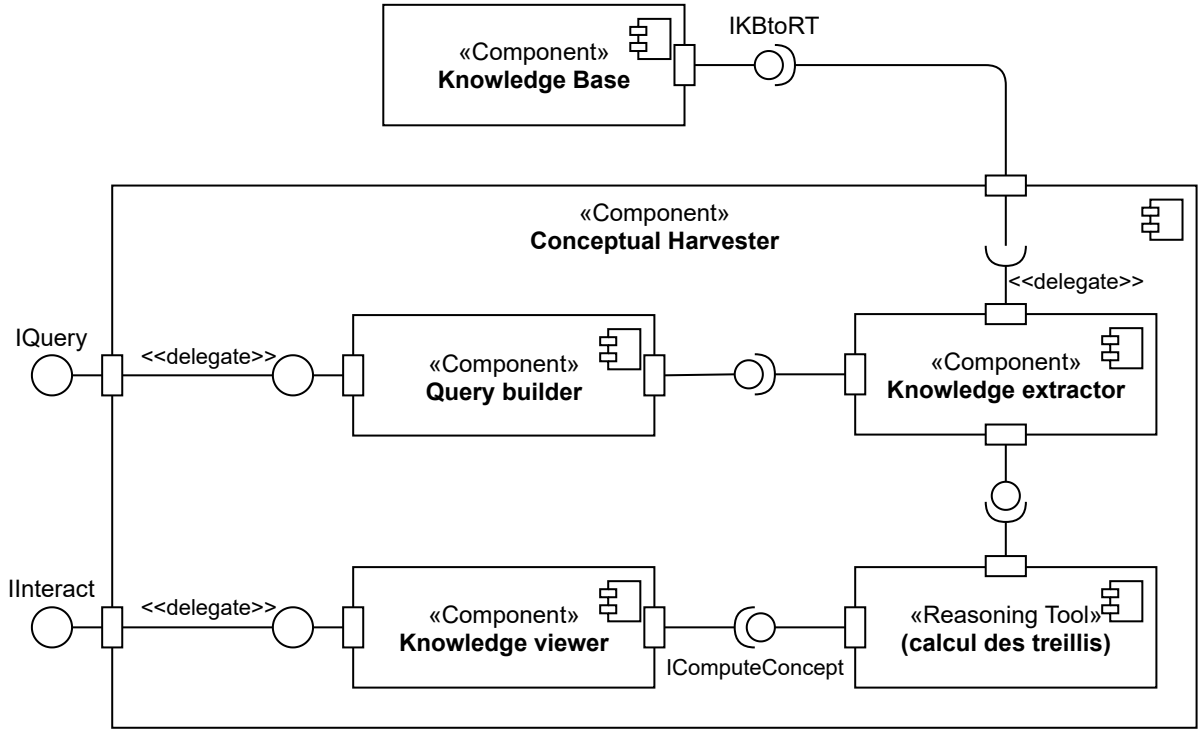


FIGURE 6.1 – *Diagramme de composants, au format UML, proposé pour supporter le système à base de connaissances Knomana. Adapté de [Keip et al., 2019b].*

plusieurs autres jeux, en perspective de son exploration par l'utilisateur final. Nous nous sommes en particulier intéressés à proposer une méthode de conversion automatique du jeu de connaissances choisi par l'utilisateur. Outre la conversion, ce prototype devra permettre d'extraire les données de plusieurs bases, à la manière d'un processus ETL. Cela impliquera d'intégrer des opérations de pré-traitement des données pour les uniformiser, e.g. via des dictionnaires.

Règles d'implication Les travaux conduits en parallèle à cette thèse ont également permis d'enrichir le composant Reasoning Tool par la formulation des connaissances sous la forme de règles d'implication en complément des treillis. L'approche adoptée permet la construction de la base d'implications de Duquenne-Guigues pour une famille de contextes relationnels, en utilisant les classifications produites par l'Analyse de Concepts Relationnels [Saoud et al., 2021, Mahrach et al., 2021]. En particulier, nous nous sommes intéressés à différentes modélisations de la relation n-aire et à leur impact sur les règles d'implication produites. Les premiers résultats ont montré qu'une division des données en plusieurs contextes formels favorise la fouille des règles du fait de la réduction de la redondance de parties de connaissances [Mahrach et al., 2021]. Ces travaux ont conduit l'équipe d'accueil à poursuivre cette approche et à proposer de classer les règles entre elles, selon des motifs liés au type de connaissance, avant de les livrer à l'expert [Saoud et al., 2021].

Visualisation Parmi les autres travaux conduits en parallèle à cette thèse, le logiciel de navigation dans une famille de treillis RCAviz [5] a été développé par un stagiaire de Master

5. http://rcaviz.lirmm.fr/

2e année, Emile Muller. Après avoir sélectionné des objets ou des attributs, le logiciel affiche le concept formel correspondant, puis permet à l'utilisateur de naviguer dans les concepts voisins de la structure conceptuelle à laquelle appartient le concept, ou dans ceux des structures conceptuelles associées via les attributs relationnels. Un historique permet de retracer l'itinéraire de navigation. RCAviz est un outil pertinent pour le système à base de connaissance, qui pourrait être inséré dans le composant Knowledge viewer.

Mise en œuvre bout-en-bout et évaluation Ce travail est la pierre angulaire du système à base de connaissances Knomana. La perspective de cet outil est de permettre la mobilisation de l'intégralité des jeux de données de Knomana, que cela concerne par exemple les plantes à effet pesticide ou antibiotique, les chaînes trophiques, ou encore les plantes toxiques utilisées pour la pêche, seules ou combinées pour répondre à des enjeux de l'agroécologie et de l'approche One Health.

Bibliographie

[Abu-Naser and Hilles, 2016] Abu-Naser, S. and Hilles, M. (2016). An expert system for shoulder problems using CLIPS. *world wide journal of Multidisciplinary Research and Development*, 2:1–8. 3

[Ackoff, 1989] Ackoff, R. L. (1989). From data to wisdom. *Journal of applied systems analysis*, 16(1):3–9. 2

[Baader et al., 2017] Baader, F., Horrocks, I., Lutz, C., and Sattler, U. (2017). *An Introduction to Description Logic*. Cambridge University Press. 24

[Bachimont, 2000] Bachimont, B. (2000). Engagement sémantique et engagement ontologique: conception et réalisation d'ontologies en ingénierie des connaissances. *Ingénierie des connaissances: évolutions récentes et nouveaux défis*, pages 305–323. 45

[Baclawski et al., 2002] Baclawski, K., Kokar, M. M., Kogut, P. A., Hart, L., Smith, J. E., Letkowski, J., and Emery, P. (2002). Extending the Unified Modeling Language for ontology development. *Software and Systems Modeling*, 1(2):142–156. 25

[Baskauf and Webb, 2014] Baskauf, S. and Webb, C. (2014). Darwin SW: Semantic Web Terms for Biodiversity Data, Based on Darwin CoreRetrieved. *March 01*. from https://github.com/darwin-sw/. 3

[Bazin et al., 2019] Bazin, A., Carbonnel, J., Huchard, M., Kahn, G., Keip, P., and Ouzerdine, A. (2019). On-demand Relational Concept Analysis. In *Proc. of ICFCA'19 (to appear)*. 23

[Bechhofer et al., 2004] Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D., Patel-Schneijder, P., and Stein, L. A. (2004). OWL Web Ontology Language Reference. Recommendation, World Wide Web Consortium (W3C). See http://www.w3.org/TR/owl-ref/. 25

[Berry et al., 2014] Berry, A., Gutierrez, A., Huchard, M., Napoli, A., and Sigayret, A. (2014). Hermes: a simple and efficient algorithm for building the AOC-poset of a binary relation. *Ann. Math. Artif. Intell.*, 72(1-2):45–71. 90

[Bertet et al., 2018] Bertet, K., Demko, C., Viaud, J., and Guérin, C. (2018). Lattices, closures systems and implication bases: A survey of structural aspects and algorithms. *Theor. Comput. Sci.*, 743:93–109. 104

[Birhanie and Tegegne, 2019] Birhanie, W. and Tegegne, T. (2019). Knowledge Based System for Diagnosis and Treatment of Mango Diseases. In Mekuria, F., Nigussie, E., and Tegegne, T., editors, *Information and Communication Technology for Development for Africa - Second International Conference, ICT4DA 2019, Bahir Dar, Ethiopia, May 28-30, 2019, Revised Selected Papers*, volume 1026 of *Communications in Computer and Information Science*, pages 11–23. Springer. 3

[Bobrow and Winograd, 1977] Bobrow, D. G. and Winograd, T. (1977). On Overview of KRL, a Knowledge Representation Language. *Cogn. Sci.*, 1(1):3–46. 24

[Boeke et al., 2004] Boeke, S., Barnaud, C., van Loon, J., Kossou, D., Huis, A., and Dicke, M. (2004). Efficacy of plant extracts against the cowpea beetle, Callosobruchus maculatus. *International Journal of Pest Management*, 50:251–258. 16

[Boucher et al., 2015] Boucher, A., Silvie, P., Menozzi, P., Adda, C., Auzoux, S., Jean, J., and Huat, J. (2015). Landscape changes in a lowland in benin : ecological impact on pests and natural enemies. *Communications in agricultural and applied biological sciences*, 80(2):79–89. 12

[Bouihi and Bahaj, 2016] Bouihi, B. and Bahaj, M. (2016). Building an e-learning system's owl ontology by exploring the uml model. 87:380–387. 46, 48

[Bouihi and Bahaj, 2019] Bouihi, B. and Bahaj, M. (2019). An UML to OWL based approach for extracting Moodle's Ontology for Social Network Analysis. *Procedia Computer Science*, 148:313–322. The Second International Conference On Intelligent Computing in Data Sciences, ICDS2018. 46, 48, 49, 50

[Brachman and Schmolze, 1985] Brachman, R. J. and Schmolze, J. G. (1985). An Overview of the KL-ONE Knowledge Representation System. *Cogn. Sci.*, 9(2):171–216. 24

[Braud et al., 2021] Braud, A., Dolques, X., Gutierrez, A., Huchard, M., Keip, P., Le Ber, F., Martin, P., Nica, C., and Silvie, P. (2021). Dealing with Large Volumes of Complex Relational Data using Relational Concept Analysis. In Missaoui, R., Kwuida, L., and Abdessalem, T., editors, *Complex Data Analytics with Formal Concept Analysis*. Springer. To appear. 20

[Calvanese et al., 2021] Calvanese, D., Gal, A., Haba, N., Lanti, D., Montali, M., Mosca, A., and Shraga, R. (2021). Adamap: Automatic alignment of relational data sources using mapping patterns. In Rosa, M. L., Sadiq, S. W., and Teniente, E., editors, *Advanced Information Systems Engineering - 33rd International Conference, CAiSE 2021, Melbourne, VIC, Australia, June 28 - July 2, 2021, Proceedings*, volume 12751 of *Lecture Notes in Computer Science*, pages 193–209. Springer. 45, 46, 48, 49, 50

[Carvalho and Santos, 2016] Carvalho, I. T. and Santos, L. (2016). Antibiotics in the aquatic environments: A review of the European scenario. *Environment International*, 94:736–757. 1

[Cellier et al., 2008] Cellier, P., Ferré, S., Ridoux, O., and Ducassé, M. (2008). A Parameterized Algorithm to Explore Formal Contexts with a Taxonomy. *Int. J. Found. Comput. Sci.*, 19(2):319–343. 48, 49, 50

[Chein and Mugnier, 2009] Chein, M. and Mugnier, M. (2009). *Graph-based Knowledge Representation - Computational Foundations of Conceptual Graphs*. Advanced Information and Knowledge Processing. Springer. 24

[Cherfaoui et al., 2020] Cherfaoui, Y., Martial, J., Morel, A., and Muller, E. (2020). Knomana - Extraction d'information dans une base de connaissances sur la santé par les plantes. *Rapport de TER, Parcours DECOL, Univ. Montpellier*. 31p. 17

[Coetzer et al., 2014] Coetzer, W., Moodley, D., and Gerber, A. (2014). A knowledge-based system for discovering ecological interactions in biodiversity data-stores of heterogeneous specimen-records: A case-study of flower-visiting ecology. *Ecol. Informatics*, 24:47–59. 3, 4

[Dammann, 2019] Dammann, O. (2019). Data, Information, Evidence, and Knowledge:: A Proposal for Health Informatics and Data Science. *Online journal of public health informatics*, 10:e224. 2

[de la Vega et al., 2020] de la Vega, A., García-Saiz, D., Zorrilla, M. E., and Sánchez, P. (2020). Lavoisier: A DSL for increasing the level of abstraction of data selection and formatting in data mining. *J. Comput. Lang.*, 60:100987. 44, 48, 49, 50

[De Silva and Wickramarachchi, 1998] De Silva, C. and Wickramarachchi, N. (1998). Knowledge-based supervisory control system of a fish processing workcell; Part II: Implementation and evaluation. *Engineering Applications of Artificial Intelligence*, 11(1):119–134. 2

[Dhar, 2013] Dhar, V. (2013). Data science and prediction. *Commun. ACM*, 56(12):64–73. 44

[Diboune et al., 2020] Diboune, S., Aougbi, L., and Kacioui, A. (2020). Extraction d'information dans une base de connaissances sur la Santé par les plantes. *Rapport de TER, Parcours IPS, Univ. Montpellier*. 44p. 22

[Dicky et al., 1995] Dicky, H., Dony, C., Huchard, M., and Libourel, T. (1995). ARES, Adding a class and REStructuring Inheritance Hierarchy. In *Onzièmes Journées Bases de Données Avancées, Nancy, France (Informal Proceedings).*, pages 25–42. 30

[Dolques et al., 2019] Dolques, X., Braud, A., Huchard, M., and Le Ber, F. (2019). RCAexplore, a FCA based Tool to Explore Relational Data. In Cristea, D., Le Ber, F., Missaoui, R., Kwuida, L., and Sertkaya, B., editors, *Supplementary Proceedings of ICFCA 2019 Conference and Workshops, Frankfurt, Germany, June 25-28, 2019*, volume 2378 of *CEUR Workshop Proceedings*, pages 55–59. CEUR-WS.org. 23

[Dolques et al., 2013a] Dolques, X., Le Ber, F., and Huchard, M. (2013a). AOC-Posets: a Scalable Alternative to Concept Lattices for Relational Concept Analysis. In Ojeda-Aciego, M. and Outrata, J., editors, *Proceedings of the Tenth International Conference on Concept Lattices and Their Applications, La Rochelle, France, October 15-18, 2013*, volume 1062 of *CEUR Workshop Proceedings*, pages 129–140. CEUR-WS.org. 30

[Dolques et al., 2013b] Dolques, X., Le Ber, F., Huchard, M., and Nebut, C. (2013b). Analyse Relationnelle de Concepts pour l'exploration de données relationnelles. In

Vrain, C., Péninou, A., and Sèdes, F., editors, *Extraction et gestion des connaissances (EGC'2013), Actes, 29 janvier - 01 février 2013, Toulouse, France*, volume RNTI-E-24 of *Revue des Nouvelles Technologies de l'Information*, pages 121–132. Hermann-Éditions. 23

[Elqassas and Abu-Naser, 2018] Elqassas, R. and Abu-Naser, S. (2018). Expert System for the Diagnosis of Mango Diseases. *International Journal of Engineering Research*, 2:10–18. 3

[Fayyad et al., 1996] Fayyad, U. M., Piatetsky-Shapiro, G., and Smyth, P. (1996). The KDD Process for Extracting Useful Knowledge from Volumes of Data. *Commun. ACM*, 39(11):27–34. 2, 5, 53

[Ferré and Cellier, 2016] Ferré, S. and Cellier, P. (2016). Graph-FCA in Practice. In *Int Conference on Conceptual Structures (ICCS 2016), Annecy, France*, volume 9717 of *Lecture Notes in Computer Science*, pages 107–121. Springer. 36

[Fowler, 2012] Fowler, M. (2012). *Patterns of Enterprise Application Architecture: Pattern Enterpr Applica Arch.* Addison-Wesley. 45

[Fowler, 2018] Fowler, M. (2018). *Refactoring - Improving the Design of Existing Code - Second Edition.* Addison Wesley object technology series. Addison-Wesley. 43, 44, 45, 48, 49, 50

[Frank, 2008] Frank, D. (2008). One world, one health, one medicine. *The Canadian veterinary journal. La revue vétérinaire canadienne*, 49:1063–5. 1

[Ganter and Kuznetsov, 2001] Ganter, B. and Kuznetsov, S. O. (2001). Pattern Structures and Their Projections. In *9th Int. Conference ICCS'01, Stanford, CA, USA*, pages 129–142. 5

[Ganter and Wille, 1999] Ganter, B. and Wille, R. (1999). *Formal Concept Analysis - Mathematical Foundations.* Springer. 4, 23, 27, 37, 38, 47, 48, 49, 50

[Garner, 1995] Garner, S. (1995). WEKA: The Waikato Environment for Knowledge Analysis. *Proceedings of the New Zealand Computer Science Research Students Conference.* 23

[Glimm et al., 2014] Glimm, B., Horrocks, I., Motik, B., Stoilos, G., and Wang, Z. (2014). Hermit: An OWL 2 reasoner. *J. Autom. Reason.*, 53(3):245–269. 3

[Hacene et al., 2013] Hacene, M. R., Huchard, M., Napoli, A., and Valtchev, P. (2013). Relational concept analysis: mining concept lattices from multi-relational data. *Ann. Math. Artif. Intell.*, 67(1):81–108. 5, 20, 23, 32

[HLPE, 2019] HLPE (2019). Agroecological and other innovative approaches for sustainable agriculture and food systems that enhance food security and nutrition. *A report by the High Level Panel of Experts on Food Security and Nutrition of the Committee on World Food Security, Rome.* 163p. 1

[Ho, 1995] Ho, T. K. (1995). Random decision forests. In *Third International Conference on Document Analysis and Recognition, ICDAR 1995, August 14 - 15, 1995, Montreal, Canada. Volume I*, pages 278–282. IEEE Computer Society. 19

[IBM, 1993] IBM (1993). *IBM Dictionary of Computing*. McGraw-Hill, Inc., USA, 10th edition. 16

[Jones and Song, 2000] Jones, T. H. and Song, I. (2000). Binary equivalents of ternary relationships in entity-relationship modeling: A logical decomposition approach. *J. Database Manag.*, 11(2):12–19. 43, 48, 49, 50, 111

[Kaytoue et al., 2011] Kaytoue, M., Kuznetsov, S. O., and Napoli, A. (2011). Revisiting Numerical Pattern Mining with Formal Concept Analysis. In Walsh, T., editor, *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 1342–1347. IJCAI/AAAI. 38

[Keip et al., 2019a] Keip, P., Gutierrez, A., Huchard, M., Le Ber, F., Sarter, S., Silvie, P., and Martin, P. (2019a). Effects of Input Data Formalisation in Relational Concept Analysis for a Data Model with a Ternary Relation. In Cristea, D., Le Ber, F., and Sertkaya, B., editors, *Formal Concept Analysis*, Lecture Notes in Computer Science, pages 191–207. Springer International Publishing. 20, 95, 104

[Keip et al., 2019b] Keip, P., Ouzerdine, A., Huchard, M., Silvie, P., and Martin, P. (2019b). Navigation conceptuelle dans une base de connaissances sur l'usage des plantes en santé animale et végétale. In Bellot, P. and Soulier, L., editors, *COnférence en Recherche d'Informations et Applications - CORIA 2019, 16th French Information Retrieval Conference. Proceedings.* ARIA. 4, 22, 23, 113

[Knublauch and Kontokostas, 2017] Knublauch, H. and Kontokostas, D. (2017). Shapes Constraint Language (SHACL). W3C recommendation, W3C. https://www.w3.org/TR/2017/REC-shacl-20170720/. 3

[Kogut et al., 2002] Kogut, P. A., Cranefield, S., Hart, L., Dutra, M., Baclawski, K., Kokar, M. M., and Smith, J. E. (2002). UML for ontology development. *Knowledge Eng. Review*, 17(1):61–64. 25

[Koona and Dorn, 2005] Koona, P. and Dorn, S. (2005). Extracts from Tephrosia vogelii for the protection of stored legume seeds against damage by three bruchid species. *Annals of Applied Biology*, 147:43 – 48. 16

[Kumar et al., 2018] Kumar, M. S., Renumol, V., and Balakrishnan, K. (2018). Design and development of a knowledge-based system for diagnosing diseases in banana plants. In *Advances in Machine Learning and Data Science*, pages 239–250. Springer. 3

[Lehmann and Wille, 1995] Lehmann, F. and Wille, R. (1995). A Triadic Approach to Formal Concept Analysis. In Ellis, G., Levinson, R., Rich, W., and Sowa, J. F., editors, *Conceptual Structures: Applications, Implementation and Theory, Third International Conference on Conceptual Structures, ICCS '95, Santa Cruz, California, USA, August 14-18, 1995, Proceedings*, volume 954 of *Lecture Notes in Computer Science*, pages 32–43. Springer. 35

[Levenshtein, 1966] Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710. Doklady Akademii Nauk SSSR, V163 No4 845-848 1965. 17

[Lieberman, 2016] Lieberman, H. (2016). Symbolic vs. Subsymbolic AI. MIT Media Lab, http://futureai.media.mit.edu/wp-content/uploads/sites/40/2016/02/Symbolic-vs.-Subsymbolic.pptx_.pdf. 19

[Loh, 2014] Loh, W.-Y. (2014). Fifty years of classification and regression trees. *International Statistical Review*, 82(3):329–348. 20

[Mahrach, 2019] Mahrach, L. (2019). Contribution à la classification de la base de connaissances knomana par les methodes ACF/ACR. *Stage de Master 1, parcours SNS, Univ. Montpellier, du 15/04 au 20/06/19*. 24p. 17

[Mahrach, 2020] Mahrach, L. (2020). Extraction de règles d'une base de connaissances du vivant classée par l'Analyse de Concepts Relationnels. *Stage de Master 2, parcours BCD, Univ Montpellier, stage du 02/03 au 14/08/ 20*. 43p. 22

[Mahrach et al., 2021] Mahrach, L., Gutierrez, A., Huchard, M., Keip, P., Marnotte, P., Silvie, P., and Martin, P. (2021). Combining Implications and Conceptual Analysis to Learn from a Pesticidal Plant Knowledge Base. In Braun, T., Gehrke, M., Hanika, T., and Hernandez, N., editors, *Graph-Based Representation and Reasoning - 26th International Conference on Conceptual Structures, ICCS 2021, Virtual Event, September 20-22, 2021, Proceedings*, volume 12879 of *Lecture Notes in Computer Science*, pages 57–72. Springer. 7, 21, 22, 113

[Mahrach et al., 2020] Mahrach, L., Gutierrez, A., Huchard, M., Keip, P., Silvie, P., and Martin, P. (2020). Extraction of association implication rules from knowledge on plants with pesticidal and antibiotic effect classified by FCA within the One-Health initiative. Journées Ouvertes de Biologie, Informatique et Mathématique (JOBIM 2020). Poster. 21, 22

[Marone, 2018] Marone, J. (2018). Formalisation d'une base de connaissances sur les plantes à effet pesticide en vue de les classifier par les méthodes AFC/ARC. *Rapport de stage de Master 1, Cirad, Montpellier*. 21p. 14, 15

[Martin et al., 2020] Martin, P., Silvie, P., Marnotte, P., and Goebel, F.-R. (2020). A Decision Support System for Determining Sugarcane Pest Reservoir. *Sugar Tech*, 22. 12

[Mbodj, 2020] Mbodj, A. (2020). Développement d'un requêteur multidimensionnel de connaissances sur la santé animale et végétale. *Rapport de stage de Master 1 SNS, Univ Montpellier, stage du 14/04 au 17/06/20*. 24p. 22

[Merwe et al., 2004] Merwe, D. v. d., Obiedkov, S. A., and Kourie, D. G. (2004). AddIntent: A New Incremental Algorithm for Constructing Concept Lattices. In *2nd Int. Conference ICFCA 2004, Sydney, Australia*, pages 372–385. 90

[Miralles et al., 2015] Miralles, A., Molla, G., Huchard, M., Nebut, C., Deruelle, L., and Derras, M. (2015). Class Model Normalization - Outperforming Formal Concept Analysis Approaches with AOC-posets. In Yahia, S. B. and Konecny, J., editors, *Proceedings of the Twelfth International Conference on Concept Lattices and Their Applications, Clermont-Ferrand, France, October 13-16, 2015*, volume 1466 of *CEUR Workshop Proceedings*, pages 111–122. CEUR-WS.org. 104

[Munzner, 2014] Munzner, T. (2014). *Visualization Analysis and Design*. A K Peters/CRC Press. 24

[Neznanov et al., 2013] Neznanov, A. A., Ilvovsky, D. A., and Kuznetsov, S. O. (2013). FCART: A New FCA-based System for Data Analysis and Knowledge Discovery. In *Contributions to the 11th International Conference on Formal Concept Analysis, Dresden, Germany, May 21-24, 2013*, pages 65–78. 47, 48, 49, 50

[Nica et al., 2016] Nica, C., Braud, A., Dolques, X., Huchard, M., and Le Ber, F. (2016). Exploring Temporal Data Using Relational Concept Analysis: An Application to Hydroecological Data. In Huchard, M. and Kuznetsov, S. O., editors, *Proceedings of the Thirteenth International Conference on Concept Lattices and Their Applications, Moscow, Russia, July 18-22, 2016*, volume 1624 of *CEUR Workshop Proceedings*, pages 299–311. CEUR-WS.org. 20

[Nolack Fote et al., 2020] Nolack Fote, F., Roukh, A., Saïd, M., Mahmoudi, S., and Debauche, O. (2020). Toward a Big Data Knowledge-Base Management System for Precision Livestock Farming. *Procedia Computer Science*, 177:136–142. 4

[Noy and Rector, 2006] Noy, N. and Rector, A. (2006). Defining N-ary Relations on the Semantic Web. W3C note, W3C. https://www.w3.org/TR/2006/NOTE-swbp-n-aryRelations-20060412/. 43, 48, 49, 50, 68

[Object Management Group (OMG), 2017] Object Management Group (OMG) (2017). Unified Modeling Language, version 2.5.1. OMG Document Number formal/2017-12-05 (https://www.omg.org/spec/UML/2.5.1/About-UML/). 54

[O'Neill, 2016] O'Neill, J. (2016). *Tackling drug-resistant infections globally: final report and recommendations*. Review on Antimicrobial Resistance. Wellcome Trust and the Department of Health of United Kingdom. 80 pages. 1

[Piatetsky-Shapiro, 1991] Piatetsky-Shapiro, G. (1991). Discovery, Analysis, and Presentation of Strong Rules. In Piatetsky-Shapiro, G. and Frawley, W. J., editors, *Knowledge Discovery in Databases*, pages 229–248. AAAI/MIT Press. 20

[Poulos et al., 2005] Poulos, M., Papavlasopoulos, S., and Chrissikopoulos, V. (2005). A Text Categorization Technique based on a Numerical Conversion of a Symbolic Expression and an Onion Layers Algorithm. *J. Digit. Inf.*, 6(1). 19

[Puckett, 2011] Puckett, J. (2011). *Zotero: A Guide for Librarians, Researchers, and Educators*. Association of College & Research Libraries. 12

[Rees, 2014] Rees, T. (2014). Taxamatch, an Algorithm for Near ('Fuzzy') Matching of Scientific Names in Taxonomic Databases. *PloS one*, 9:e107510. 17

[Reverter et al., 2020] Reverter, M., Tapissier-Bontemps, N., Sarter, S., Sasal, P. P., and Caruso, D. (2020). Moving towards more sustainable aquaculture practices: a meta-analysis on the potential of plant-enriched diets to improve fish growth, immunity and disease resistance. *Reviews in Aquaculture*. 1

[Rodríguez-García and García-Sánchez, 2020] Rodríguez-García, M. Á. and García-Sánchez, F. (2020). CropPestO: An Ontology Model for Identifying and Managing Plant Pests and Diseases. In Valencia-García, R., Alcaraz-Mármol, G., del Cioppo-Morstadt, J., Vera-Lucio, N., and Bucaram-Leverone, M., editors, *Technologies and Innovation - 6th International Conference, CITI 2020, Guayaquil, Ecuador, November 30 - December 3, 2020, Proceedings*, volume 1309 of *Communications in Computer and Information Science*, pages 18–29. Springer. 3

[Rodríguez-García et al., 2021] Rodríguez-García, M., Garcia-Sanchez, F., and Valencia-García, R. (2021). Knowledge-Based System for Crop Pests and Diseases Recognition. *Electronics*, 10:905. 3, 4

[Rosa et al., 2021] Rosa, M. D., Fenza, G., Gallo, A., Gallo, M., and Loia, V. (2021). Pharmacovigilance in the era of social media: Discovering adverse drug events cross-relating Twitter and PubMed. *Future Gener. Comput. Syst.*, 114:394–402. 20

[Rossi et al., 2006] Rossi, F., van Beek, P., and Walsh, T., editors (2006). *Handbook of Constraint Programming*, volume 2 of *Foundations of Artificial Intelligence*. Elsevier. 20

[Rowley, 2007] Rowley, J. (2007). The wisdom hierarchy: Representations of the DIKW hierarchy. *J Inf Sci*, 33. 2

[Rumbaugh et al., 2004] Rumbaugh, J., Jacobson, I., and Booch, G. (2004). *Unified Modeling Language Reference Manual, The (2Nd Edition)*. Pearson Higher Education. 25

[Samb, 2019] Samb, M. (2019). Développement d'une interface de Navigation pour la base de connaissances Knomana. *Stage de Master 1, parcours BCD, Univ. Montpellier, du 23/04 au 26/06/19*. 26p. 22

[Saoud et al., 2021] Saoud, J., Gutierrez, A., Huchard, M., Marnotte, P., Silvie, P., and Martin, P. (2021). Explicit versus Tacit Knowledge in Duquenne-Guigues Basis of Implications: Preliminary Results. 113

[Schmidhuber, 2015] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117. 19

[Sebastian, 2021] Sebastian, R. (2021). Formal concept analysis - contexts, concepts and concept lattices. 37

[Silvie et al., 2021] Silvie, P., Martin, P., Huchard, M., Keip, P., Gutierrez, A., and Sarter, S. (2021). Prototyping a Knowledge-Based System to Identify Botanical Extracts for Plant Health in Sub-Saharan Africa. *Plants*, 10:896. 4, 11, 13

[Skobelev et al., 2019] Skobelev, P., Simonova, E., Smirnov, S., Budaev, D., Voschuk, G., and Morokov, A. (2019). Development of a Knowledge Base in the "Smart Farming" System for Agricultural Enterprise Management. *Procedia Computer Science*, 150:154–161. 4

[Skoutas et al., 2009] Skoutas, D., Simitsis, A., and Sellis, T. (2009). *Ontology-Driven Conceptual Design of ETL Processes Using Graph Transformations*, pages 120–146. Springer Berlin Heidelberg, Berlin, Heidelberg. 42, 48, 49, 50

[Sowa, 1984] Sowa, J. F. (1984). *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley. 24

[Sowa, 1991] Sowa, J. F., editor (1991). *Principles of Semantic Networks - Explorations in the Representation of Knowledge*. The Morgan Kaufmann Series in representation and reasoning. Morgan Kaufmann. 24

[Stumme et al., 2002] Stumme, G., Taouil, R., Bastide, Y., Pasquier, N., and Lakhal, L. (2002). Computing iceberg concept lattices with Titanic. *Data & Knowledge Engineering*, 42(2):189 – 222. 30, 91

[Sung and Chi, 2020] Sung, H. and Chi, Y. (2020). A knowledge-based system to find over-the-counter medicines for self-medication. *J. Biomed. Informatics*, 108:103504. 4

[Sunyé et al., 2001] Sunyé, G., Pollet, D., Traon, Y. L., and Jézéquel, J. (2001). Refactoring UML Models. In Gogolla, M. and Kobryn, C., editors, *«UML» 2001 - The Unified Modeling Language, Modeling Languages, Concepts, and Tools, 4th International Conference, Toronto, Canada, October 1-5, 2001, Proceedings*, volume 2185 of *Lecture Notes in Computer Science*, pages 134–148. Springer. 44, 48, 49, 50

[Teorey et al., 1986] Teorey, T., Yang, D., and Fry, J. (1986). A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model. *ACM Comput. Surv.*, 18:197–222. 25

[Trujillo and Luján-Mora, 2003] Trujillo, J. and Luján-Mora, S. (2003). A UML Based Approach for Modeling ETL Processes in Data Warehouses. In Song, I., Liddle, S. W., Ling, T. W., and Scheuermann, P., editors, *Conceptual Modeling - ER 2003, 22nd International Conference on Conceptual Modeling, Chicago, IL, USA, October 13-16, 2003, Proceedings*, volume 2813 of *Lecture Notes in Computer Science*, pages 307–320. Springer. 42, 48, 49, 50

[Vo and Hoang, 2020] Vo, M. H. L. and Hoang, Q. (2020). Transformation of UML class diagram into OWL Ontology. *J. Inf. Telecommun.*, 4(1):1–16. 25

[Voutsadakis, 2002] Voutsadakis, G. (2002). Polyadic Concept Analysis. *Order*, 19(3):295–304. 102

[W3C, 2014] W3C (2014). Resource Description Framework (RDF) Schema. 25

[Wako, 2017] Wako, D. A. (2017). Development of Knowledge Based System for Wheat Disease Diagnosis: A Rule Based Approach. *Journal of Computer Engineering and Information Technology*, 06. 2

[Wang and Chan, 2001] Wang, X. and Chan, C. W. (2001). Ontology Modeling Using UML. In Wang, X., Johnston, R., and Patel, S., editors, *OOIS 2001*, pages 59–68, London. Springer London. 25

[Wolfert et al., 2017] Wolfert, S., Ge, L., Verdouw, C., and Bogaardt, M. (2017). Big Data in Smart Farming – A review. *Agricultural Systems*, 153:69–80. 4

[Wood et al., 2014] Wood, D., Lanthaler, M., and Cyganiak, R. (2014). RDF 1.1 Concepts and Abstract Syntax. W3C recommendation, W3C. 45

[Yelapure, 2017] Yelapure, S. J. (2017). Knowledge Based System Using Fuzzy Logic for Soybean Crop with Special Reference to Nutrition Management. *International Journal for Research in Applied Science and Engineering Technology*, pages 1660–1669. 2