l'université nantes angers le mans
COMMUNAUTÉ D'UNIVERSITÉS ET ÉTABLISSEMENTS

UNIVERSITÉ DE NANTES

Thèse de Doctorat

Frédéric
DUMONCEAUX

Mémoire présenté en vue de l'obtention du
grade de Docteur de l'Université de Nantes
sous le label de l'Université de Nantes Angers Le Mans

École doctorale : Sciences et technologies de l'information, et mathématiques

Discipline : Informatique et applications, section CNU 27
Unité de recherche : Laboratoire d'informatique de Nantes-Atlantique (LINA)

Soutenue le 16 Octobre 2015

Approches algébriques pour la gestion et l'exploitation de partitions sur des jeux de données

JURY

Rapporteurs :	**M. Sofian MAABOUT**, Maître de Conférences (HDR), Université de Bordeaux 1 **M^{me} Marie-Christine ROUSSET**, Professeur, Université de Grenoble
Examinateurs :	**M. Amedeo NAPOLI**, Directeur de recherche, CNRS, Université Henri-Poincaré (Nancy 1) **M. Jin-Kao HAO**, Professeur, Université d'Angers
Directeur de thèse :	**M. Marc GELGON**, Professeur, Université de Nantes

Remerciements

Bien que je n'aie jamais prêté d'attention particulière à la section des remerciements dans les manuscrits que j'ai pu parcourir lors de ma thèse de doctorat, je me suis posé la question de sa pertinence dans le récit global de ma thèse et du formalisme qui y est développé.

Cette thèse n'est pas simplement l'aboutissement de plusieurs années d'un travail entrepris il y a quatre ans, mais plutôt le prolongement d'une vocation qui est apparue lors de mes études universitaires et que je dois aux personnes rencontrées durant toutes ces années.

J'ai une pensée toute particulière pour Alexandre Dikovsky, logicien respecté, décédé en 2014, qui m'enseigna les rudiments de la logique et du raisonnement pendant mes années de licence, puis d'intelligence artificielle en master. Déconcerté par le personnage et l'étendue de ses connaissances, il était souvent difficile de s'imprégner de sa compréhension des problèmes auxquels il nous confrontait. Son matériel pédagogique était également impénétrable du fait de la rigueur nécessaire à sa compréhension. Pourtant, il faisait preuve d'une grande patience ; il exprimait toujours la volonté de nous aider à surmonter nos difficultés et n'était jamais avare s'agissant de conseils pratiques ou de lectures permettant d'approfondir certains points.

C'est de cette expérience initiale qu'est né mon engouement pour la recherche scientifique, la volonté d'approfondir mes connaissances et ma réconciliation avec l'usage des mathématiques. D'autres rencontres avec des enseignants-chercheurs ont également été déterminantes dans ma volonté de réaliser un doctorat et mes remerciements pour l'aide et le soutien qu'ils m'ont apportés leur sont également adressés.

Cette thèse n'aurait évidemment pas existé sans le concours de mes encadrants Marc Gelgon et Guillaume Raschia qui m'ont offert cette opportunité. Tous les deux ont été présents tout le long de ma thèse pour me conseiller sur le plan scientifique et humain. En particulier, ils ont toujours su me diriger tout en me permettant de travailler en autonomie au jour le jour et en me laissant le choix des pistes à suivre. Leur aide dans l'écriture de mes articles, ainsi que le présent manuscrit, fut également d'une aide précieuse pour communiquer mes résultats. Je les remercie chaleureusement de leur soutien et de leur présence pendant ces quatre années.

Je remercie également Marie-Christine Rousset et Sofian Maabout qui m'ont fait l'honneur d'être les rapporteurs de ma thèse ainsi que Jin-Kao Hao pour avoir accepté

d'être examinateur de celle-ci. En outre, je remercie Amedeo Napoli pour sa présence en tant que président du jury malgré ses nombreux engagements professionnels pendant la même période.

Je tiens également à remercier l'ensemble des membres des anciennes équipes GRIM et COD, constituant désormais l'équipe DUKe, ainsi que les secrétaires du département pour l'ambiance de travail chaleureuse et décontractée autant sur le plan de la recherche que de l'enseignement, ainsi que les discussions passionnantes et passionnées devant la machine à café ou au détour d'un couloir et bien sûr en salle de travaux pratiques.

Dernier point, mais non des moindres, je tiens à remercier mes amis et mes proches, ainsi que ceux qui le sont devenus pendant ces quatre années, et qui ont été un soutien de poids dans les moments de doute et ont su trouver les mots justes pour surmonter les passages à vide.

Table des matières

1

Introduction

Les deux dernières décennies ont vu l'essor des méthodes d'analyses de données multidimensionnelles dans des domaines d'application aussi variés que les sciences humaines, la sociologie ou l'économie [124, 24, 116]. Le principe général à l'oeuvre est fondé sur la recherche d'une ou plusieurs informations statistiques permettant de résumer l'information afin d'établir un caractère d'homogénéité, s'il en existe, et donc de faciliter l'analyse par une tierce personne. La transformation de ou des informations vers un savoir intelligible n'en demeure pas moins une tâche peu évidente, car elle impose une altération de celles-ci sous l'influence de notre perception à déterminer ce qui est pertinent ou significatif.

Quelle que soit l'application, on souhaite pouvoir manipuler les informations extraites à l'aide de représentations symboliques des données et d'un ensemble de primitives, sous la forme de prédicats ou de relations, qui décrivent comment interagissent ces éléments entre eux. Une description ontologique permet alors de dégager l'ensemble des concepts permettant de qualifier ce qui existe en utilisant les notions du domaine. On distingue ainsi les entités et les prédicats comme étant les unités de base de la connaissance (*c.f.* [118] et les études plus récentes [61, 25]).

Les entités sont des unités ponctuelles ou discrètes qui forment naturellement l'*univers du discours*, soit le monde avec lequel on souhaite interagir ou le sujet sur lequel portent les prédicats. Par des combinaisons des prédicats avec des connecteurs logiques, on est alors en mesure de formuler des propositions logiques dont la véracité dépend des faits qui décrivent de manière abstraite l'état du monde à un instant particulier. Celles-ci sont également les propositions atomiques à partir desquelles toutes les autres sont formulables [34].

La première présentation extensive de cette doctrine est due à Russell qui l'utilise de manière explicite dans [1]. Son premier dessein, s'il en est, est de comprendre la logique qui sous-tend les mathématiques. En cela, il s'oppose à la vision métaphysique, dominante à l'époque, qui identifie tout ce qui existe à un tout unique et irréductible et dont les constituants ne sauraient être perçus que comme quantité négligeable, ce qui rend chimérique toute tentative de représentation de ces entités sur le plan épistémologique et donc sur la représentation du monde, quelles qu'en soient ses frontières supposées ou réelles.

Énoncé de la sorte, ce principe général transcende les pratiques et usages en vigueur dans la plupart des disciplines de l'informatique théorique ou appliquée. Par exemple, le *rasoir d'Ockam* ou *principe de parcimonie* chez *Bentham* qui s'énonce de manière littérale : « les entités ne doivent pas être multipliées par delà ce qui est nécessaire » est souvent utilisé dans les processus de construction du sens d'un champ de connaissances pour en déterminer un ensemble représentatif de concepts propres à raisonner sur celles-ci. Une méthodologie que l'on peut rapprocher du phénomène de surapprentissage qui advient lorsqu'un trop grand nombre d'informations est utilisé afin d'éliciter les connaissances d'un modèle (*c.f.* également [13, 122, 132]).

Le processus de conceptualisation matérialise alors ses concepts comme les éléments d'une ontologie plus spécifique que celle employée pour le décrire. À l'instar de l'ontologie de Russell, la doctrine de Bentham tend vers une interprétation rationnelle des liens entre les entités constitutives du monde ou de ses parties ; principe que l'on retrouve chez Frege, Whitehead et Russell, entre autres [119].

Suivant cette perspective, la recherche raisonnée des causes, présidant aux observations empiriques, met en avant la variété des choses qui existent indépendamment les unes des autres, des propriétés qu'elles exhibent et des relations qu'elles entretiennent.

Énoncer des vérités ou des faits est en dernier lieu dépendant d'un ensemble d'entités, qui sont soit des éléments particuliers, chacun exposant une propriété remarquable, soit une relation survenant entre plusieurs éléments particuliers. La méthodologie s'articule ensuite autour de l'accomplissement de certaines combinaisons d'entités atomiques menant à la définition d'éléments plus complexes, mais dont l'interprétation est réalisable et cohérente.

Les atomes représentent donc une information de base sur laquelle on peut agir et dotée d'une sémantique établie. De cette dialectique découle le concept de *langage* où les mots font référence à des entités particulières, leurs propriétés et relations, et les constantes logiques qui peuvent expliquer toutes les vérités, en dépit de leur vocabulaire limité. Cette conception méthodologique du raisonnement est un évènement important puisque cette approche perdure jusqu'à nos jours, indépendamment des apports conceptuels et méthodologiques réalisés au vingtième siècle par les logiciens et algébristes.

Sur le dernier point, les membres de chaque communauté ont de prime abord adopté des voies différentes. Toutes deux sont légitimes sur le plan méthodologique et leurs interactions s'avéreront fécondes et déboucheront sur de nouvelles théories. Le point de

vue qui nous intéresse ici est celui adopté par *Lindembaum* et *Tarski*.

De manière générale, l'algèbre introduite par *Boole* [21, 22] vise à trouver un fondement algébrique au calcul des propositions logiques en mettant l'accent sur les points communs et les divergences avec le calcul arithmétique. Le point crucial étant la dématérialisation des règles de calculs et des éléments – donc l'univers – sur lesquels ils s'appliquent. En cela, seules les identités remarquables, advenant entre des équations où les variables sont libres, *caractérisent* l'essence des règles logiques.

À cela, Tarski et Lindenbaum observeront [130] que le calcul des propositions peut être plongé dans une algèbre plus intéressante que l'algèbre des formules. Ils identifient une relation d'équivalence logique opérant sur des classes de formules symbolisant des *théories*. Puis, cet « artifice » va permettre de réaliser une passerelle entre le calcul des propositions et l'algèbre de Boole où les éléments sont précisément des classes de formules. Par voie de conséquence, on peut manipuler de manière transparente un représentant de chaque théorie par le biais d'un modèle.

L'ensemble ainsi obtenu est équipé des opérations idoines dont les propriétés sont généralement calquées sur les connecteurs logiques dans le but de mener un raisonnement logique. La structure associée est alors l'algèbre quotient des formules et souvent désigné par l'algèbre de Lindembaum-Tarski. Cette construction donnera naissance à la logique algébrique dont le fondement est la recherche de classes d'algèbres jouant le rôle de modèles d'interprétation pour des systèmes déductifs.

Le point qui nous semble essentiel est la nécessité d'une structuration logique comme moyen de traitement de l'information en vue de l'organiser et de la rendre la plus intelligible possible. Le second point est de rendre cette représentation aisément manipulable, à tout le moins, de ne pas modifier la manière dont sont perçues les connaissances extraites ou construites depuis les informations de base.

La structure de données transversale aux travaux que nous présentons dans cette thèse est celle de *partition d'ensemble* dont les diverses instanciations à travers ce manuscrit vont nous permettre de modéliser les propriétés relatives à des jeux de données. Cette représentation vient naturellement à l'esprit tant elle apparaît fréquemment dans la littérature, et en particulier dans les travaux portant sur la classification non supervisée et dans la construction des modèles de systèmes déductifs.

En outre, à l'instar du théorème de Cayley qui établit la propriété que tout groupe a naturellement une représentation sous la forme de permutations, les morphismes d'algèbre définissent une relation d'équivalence sur leur image respective et induisent donc un partitionnement de l'ensemble sous-jacent. Chaque partie reflète alors le résultat ou le modèle d'une opération particulière et ouvre la voie à une caractérisation de leur calcul sous ce prisme [103, 112]

Dans les applications tirant profit d'algorithmes de clustering [71, 72, 137], les modèles figurent ainsi des classifications des objets suivant un ou plusieurs critères endogènes. L'espace des points est alors similaire à l'univers des entités sur lequel on est amené à raisonner. Les modèles idéaux seront ceux où les points au sein d'une même

classe sont les plus similaires, suivant l'objectif fixé, *et* les plus dissemblables en comparaison des points appartenant aux autres classes. Cela implique également que les propriétés observables sur les points soient cohérentes au niveau de leur classe respective.

Au-delà de l'aspect qualitatif permettant l'évaluation des algorithmes produisant de tels modèles, il est généralement délicat de *résumer* des jeux de données entiers à l'aide d'un modèle global, satisfaisant simultanément à un ensemble d'objectifs dont la question de la représentativité n'est elle-même pas tranchée [53, 105].

En effet, la profusion toujours plus importante de nouvelles données, indépendamment de la croissance de la capacité de calculs et de stockage, rend délicate l'extraction de connaissances exploitables. La mise en commun de données provenant de sources différentes rend encore plus difficile l'élaboration d'une vérité de terrain sur laquelle s'appuyer et réaliser des tendances. Sur le plan opérationnel, la diversité des approches un compromis particulier accroît l'effort nécessaire afin de mettre en œuvre des solutions communes [97, 65].

Imposer une représentation sous la forme de partitions d'ensemble, instances de différents modèles, de différentes provenances, devrait faciliter l'opérationnalisation de ces résultats et valoriser les connaissances ainsi extraites en permettant leur réutilisation. Par conséquent, il est alors également nécessaire d'étudier les contextes dans lesquels ces modèles sont susceptibles d'interagir harmonieusement ; soit comment l'interprétation de ces modèles est-elle préservée lorsqu'on les combine pour produire de nouveaux résultats.

Le manuscrit se divise en deux parties indépendantes, la première étant dévolue aux notions et concepts utilisés par la suite (chapitres 2 et 3), la seconde fait état des diverses propositions que nous avons réalisées durant cette thèse (chapitres 4, 5, 6).

Chapitre 2

Le premier chapitre est une présentation assez générale de l'algèbre universelle pour laquelle nous rappellerons les principaux résultats et outils. Ceux-ci permettront d'amener à la construction des partitions d'ensembles et des propriétés afférentes à leur manipulation : les partitions renfermant des classes d'équivalence et qui sont également des *congruences*, c'est-à-dire stables pour les opérations induites.

L'intérêt de ces structures est bien connu et l'utilisation du langage abstrait de l'algèbre nous permettra de justifier leur apport dans des contextes où les informations qu'elles résument, apportent une facilité d'interprétation de celles-ci. L'usage privilégié concernera en particulier la construction des modèles dans l'esprit de l'algèbre de Lindembaum-Tarski pour la construction de modèles d'interprétation, et qui servira en partie de support à notre proposition du chapitre 4.

Chapitre 3

Le premier contexte dans lequel nous utiliserons les partitions et leurs classes sera celui des ensembles ordonnés et leur pendant algébrique, les treillis. La notion de générateur déjà introduite dans le chapitre précédent sera centrale dans celui-ci et nous permettra d'introduire certains théorèmes de représentation sur les treillis.

Cette approche permet la construction d'une structure analogue – isomorphe – et décrite à l'aide de tels éléments. Le premier apport de ces représentations est conceptuel : il permet par la décomposition des éléments d'une structure de révéler une nouvelle facette de leur combinatoire et sous certaines conditions, l'adjonction de nouvelles opérations.

La concrétisation principale d'un de ces théorèmes interviendra dans le chapitre 4 dans le cadre de l'ajout de nouveaux opérateurs agissant sur les partitions d'ensembles.

Chapitre 4

La première proposition que nous allons formuler dans ce manuscrit propose de plonger les opérations ensemblistes de l'algèbre relationnelle dans le treillis des partitions. Nous allons exploiter la dualité induite par un théorème de représentation des treillis.

Celle-ci va nous permettre d'opérationnaliser les partitions dans le but de construire des vues matérialisées de cubes de données, objet d'une publication intitulée : *Materializing Data Cubes as Partitions of Sets of Tuples* [45].

La dualité que nous allons exposer permettra d'établir un pont entre le schéma d'une relation d'une base de données, naturellement équipé d'une structure de treillis $\mathcal{P}(\mathcal{A})$ où $\mathcal{A}$ est l'ensemble des attributs et des partitions d'ensemble de Π_Ω représentant des résultats de requêtes, où Ω symbolise l'univers des tuples. Ainsi, les éléments de la relation induite $\Pi_\Omega \times \mathcal{P}(\mathcal{A})$ vont permettre de construire les modèles maximaux pour chaque sélection d'attributs, que nous nommerons *partitions annotées* et soulignant le fait qu'une partition P est l'instance associée au modèle de la requête X.

Le premier avantage découlant de cette représentation, est que les classes induites dans chaque partition annotée, sont également des relations de congruence. Il est donc possible de réaliser des opérations du treillis des partitions de sorte que ces opérations simulent le résultat d'une requête particulière représentée par les annotations associées à son modèle.

Le second avantage que procure cette méthode est d'induire des partitions sur $\mathcal{P}(\mathcal{A})$ dès lors que deux requêtes partagent exactement les mêmes modèles. Une application que nous proposerons sera d'exploiter les propriétés d'un algorithme de fermeture sur $\mathcal{P}(\mathcal{A})$ où chaque sous-ensemble X peut être représenté par le plus grand sous-ensemble maximal Y et tel que le modèle associé P_Y ait été préalablement calculé.

Un ensemble de partitions annotées figure alors la représentation complémentaire

d'une base de données, et dont nous étudierons les propriétés. Comme preuve de concept, des expérimentations sur le banc d'essai TPC-H sont également fournis afin d'illustrer notre propos.

Chapitre 5

Ce chapitre contient notre proposition relative au problème du calcul de la partition médiane. Il consiste en une version très largement étendue de l'article : *An algebraic approach to ensemble clustering* [43].

Ainsi que dans le précédent chapitre, nous traitons la question de la représentation de certains modèles à l'aide de partitions d'ensemble. Ici, les partitions représentent des résultats produits à partir d'algorithmes de clustering sur un jeu de données. Étant donné un ensemble de partitions, chacune représentant une classification suivant des critères particuliers, nous essayons de construire une partition réalisant un compromis parmi toutes les possibilités d'agrégation proposées, et reflétant fidèlement les spécificités de chaque partition en entrée.

L'approche que nous utilisons est fondée sur la recherche d'une partition médiane qui dépend de l'application d'un vote à la majorité sur l'appartenance de deux points au sein de la même classe, dès qu'une majorité d'entre elles soutient cette observation [14]. Cette méthode s'appuie elle-même sur la définition d'une relation ternaire unifiant les résultats des opérations de complémentation réalisées deux-à-deux entre chaque élément [17].

L'adoption de cette règle dans le contexte du treillis des partitions soulève plusieurs problèmes ; cette opération n'est applicable que dans les treillis vérifiant certains axiomes. En effet, l'absence d'information caractérisant le contexte logique dans lequel les partitions ont été construites, n'apporte aucune contrainte ou limite sur la manière de combiner celles-ci. De ce fait, aucune propriété n'est transportée pour contraindre le calcul entre partitions, contrairement au chapitre précédent où les modèles sont des partitions d'une *algèbre* et vérifient les contraintes associées du système qu'elles émulent.

La seconde difficulté est représentée par l'impossibilité de caractériser l'opération de complémentation dans ce contexte, dont dépend le calcul d'un élément médian.

Notre proposition est donc fondée sur l'exploration de nouvelles approches conceptuelles permettant la modélisation en dehors du cadre habituel, brièvement présenté ci-dessus. En particulier, nous allons instancier un théorème de représentation qui autorise la définition de nouveaux opérateurs au treillis des partitions, ne nécessitant pas la formulation de contraintes particulières sur les identités et axiomes de la structure.

Chapitre 6

Ce chapitre constitue notre proposition concernant l'intégration des partitions d'ensemble au sein des SGBD relationnels et d'en simuler les opérations. Nous réaliserons la simulation d'opérateurs ensemblistes et latticiels entre partitions et sur leurs classes, indépendamment du contexte dans lequel elles sont instanciées, à l'aide de l'algèbre relationnelle et de certaines extensions propres à SQL3. Une étude expérimentale est menée avec une implémentation *ad hoc* afin de déterminer le bien-fondé de l'approche.

Il compile les articles suivants :

- *Computing Partitions within SQL Queries : A Dead End ?* [46]
- *A First Attempt to Computing Generic Set Partitions : Delegation to an SQL Query Engine* [45]

Une partie de ce chapitre est également dédiée à la mise en œuvre du calcul de la borne supérieure entre deux partitions $P \vee Q$ à l'aide du cadre algorithmique Union-Find. À l'aide de cette approche, nous proposerons diverses variantes de l'algorithme afin de réaliser cette opération, permettant de choisir le plus adapté à notre modèle de données introduit précédemment.

2

Éléments d'algèbre universelle

Cette thèse débute en développant quelques éléments d'algèbre universelle, utiles pour appréhender les connections entre quelques instances remarquables de structures algébriques tels les semi-anneaux et quelques sous-ensembles. En effet, il est souvent intéressant d'extraire, quand cela est possible, des propriétés communes pouvant être partagés par différents types de structures algébriques. Cela permet de découvrir des principes généraux, des méthodes de construction et des résultats qui peuvent à un haut niveau d'abstraction, s'appliquer à de nouvelles problématiques et faire émerger de nouvelles pistes.

Parmi les structures d'intérêt, les treillis sont les structures privilégiées pour modéliser puis manipuler la plupart des calculs propositionnels et les logiques avec quantificateurs de manière opérationnelle. Les dioïdes sont eux plus généralement impliquées dans la résolution de certains systèmes d'équations liés à des problèmes de graphes. Le prochain chapitre développera ces structures et représente la base de notre travail.

Dans ce qui suit, on développe quelques idées et concepts généraux appartenant à l'*algèbre universelle* dont la notion fondamentale est évidemment celle d'une *algèbre*.

2.1 Introduction / Motivation

Traditionnellement, une algèbre est présentée comme la donnée d'un ensemble d'entités et d'opérations sur ces mêmes éléments, permettant d'en atteindre d'autres au sein du même ensemble. A cet égard, la porosité entre cette description et celle des *relations*

apparaît clairement et illustre aisément la dualité autour de ces deux concepts. À titre d'exemple, l'opération d'addition $(+)$ entre entiers naturels $\mathbb{N}$ lie entre eux certains nombres, de cela résulte alors un nouveau nombre, lui-même pouvant être librement et indéfiniment associé.

A contrario, l'existence d'une opération inverse $(-)$ semble aller de soi au premier abord, bien qu'elle soit infondée puisqu'elle transgresse une règle de base : le résultat d'une soustraction $a - b$ peut ne pas être un entier naturel. Ce constat est loin d'être une argutie même si il paraît évident d'objecter par l'emploi des entiers relatifs pour lesquels cette opération est correctement définie. En particulier, la présentation duale a le merite d'exposer ce point avec concision :

$$a \leq b \Leftrightarrow (\exists x)(a + x = b \Leftrightarrow x = b - a) \quad (\forall a, b \in \mathbb{N}) \tag{2.1}$$

où $\leq$ est entièrement définie sur $\mathbb{N} \times \mathbb{N}$. Cette dernière permet de définir l'addition comme une relation ternaire $(a, x, b) \in \mathbb{N}^3$. La projection sur le second membre projette (au sens de la surjection) chaque paire (a, b) sur l'ensemble de ses élément adjoints $\{x_i\}$.

Le résultat est défini implicitement par l'existence d'au moins un élément pouvant être adjoint[1] à a pour obtenir b. Elle engendre deux copies de $\mathbb{N}$ représentant chacun l'ensemble contenant le résultat de la soustraction selon :

$$(\forall a, b)(a - b \in \mathbb{Z}^- + \mathbb{Z}^+) \tag{2.2}$$

où :

$$\mathbb{Z}^- = \{-x \mid x \in \mathbb{N}\} \tag{2.3}$$
$$\mathbb{Z}^+ = \mathbb{N} \tag{2.4}$$

Néanmoins, cette construction opérationnelle peut ne pas être possible dans tous les ensembles concrets, par exemple, les mots formés sur un alphabet Σ n'ont pas d'inverses naturels bien que les propriétés formelles codifiant la combinaison de leurs éléments sont semblables à celles de l'arithmétique (ce sont tous deux des monoïdes et obéissent donc aux axiomes idoines). Généralement, il existe des méthodes permettant sous certaines conditions de construire des inverses ; ces derniers devant répondre aux propriétés abstraites désirées.

Cet exemple illustre l'avantage d'employer les relations afin de faciliter l'étude ou l'émergence de nouvelles propriétés algébriques et réciproquement. Par ailleurs, cette dualité est profitable dans la présentation algébrique d'une logique selon qu'on souhaite insister sur la notion d'implication (liée à l'ordre) ou sur la notion de connecteurs logiques (liée aux opérateurs).

La prochaine section sera ainsi dévolue à la définition d'une structure d'algèbre et de relation, puis de l'étude de leurs propriétés communes.

[1] On parle également de résiduation. Certains détails sont volontairement omis sur la construction et seront plus longuement abordé dans les sections traitant de l'usage opérationnel de cette méthode.

2.2 Relations et Structures

2.2.1 Préliminaires

Définition 2.2.1 (Relation)**.** *Une* relation d'arité n *ou* n*-aire définie sur un domaine ou univers mathématique* A*, modélise une propriété que certaines de ses entités peuvent vérifier. L'*extension *de la relation est l'ensemble qui contient alors les tuples* $(x_i)_{1\leq i\leq n} \in A^n$ *vérifiant cette propriété. Lorsque la définition d'une relation est une phrase ou une formule dans un langage, on parle plus spécialement de* modèle*.*

Reprenant la définition de l'ordre sur les entiers naturels 2.1, on a :

$$a \leq b \Leftrightarrow (a, b) \in \leq \ \subseteq \ \mathbb{N}^2 \tag{2.5}$$

qui est une relation 2-aire ou binaire.

Définition 2.2.2 (Structure de relations)**.** *Une* structure *est, par définition, la donnée d'un ensemble* A *muni d'une famille indexée de relations* (R_i) *sur* A*. La structure se voit également associée la famille des arités de chaque relation représentant la structure. Cette famille représente le* type *de la structure. Deux structures sont similaires si elles ont le même type.*

Par la suite, on notera indifféremment A le domaine d'une structure et la structure elle-même, en l'absence d'ambiguïté dans son usage.

Définition 2.2.3 (Opération)**.** *Par une* opération *sur un ensemble* A*, on entend une opération* n*-aire qui associe un élément de* A *à chaque* n*-uplet de* A*. Une opération est alors une fonction totale.*

L'addition entre entiers naturels est donc une opération binaire associant un entier naturel à chaque paire d'entiers naturels.

Définition 2.2.4 (Structure algébrique)**.** *Une algèbre est définit comme un ensemble muni d'opérations* n*-aire soit une structure d'opérations* (O_i) *sur* A*.*

On définit de manière identique le type d'une structure d'opérations et de relations. La notion de similarité entre algèbres en découle naturellement.

Définition 2.2.5 (Similarité)**.** *Deux structures algébriques* A *et* B *sont similaires si elles sont de même type.*

À ce titre, les opérations sur les entiers naturels $(\mathbb{N}, +, 0)$ et $(\mathbb{N}, \times, 1)$ sont similaires car ayant le même type $(2, 0)$. 0 représente dans ce cas l'arité de l'opération 0-aire (nullaire) engendrant l'*élément distingué* ou *neutre* de chaque structure.

À cet instant, la dualité exposée sur $\mathbb{N}$ (2.1) se transpose naturellement aux structures de relations et d'opérations : chaque opération O_i d'arité a_i sur une algèbre A se voit associée une relation R_i d'arité $a_i + 1$:

$$a + b = c \Leftrightarrow (a, b, c) \in + \tag{2.6}$$

On peut donc invariablement considérer une algèbre et une relation[2] comme deux interprétations équivalentes du même objet mathématique. On voit que cette définition généralise celle de graphe d'une opération $f : X \to Y$ dont voici une définition formelle.

Définition 2.2.6 (Graphe d'une opération)**.** *Soit une algèbre A muni d'une opération n-aire O,*

$$\{(a_1, \ldots, a_{n+1}) \in A^{n+1} \mid O(a_1, \ldots, a_n) = a_{n+1}\} \tag{2.7}$$

forme le graphe de O.

Lorsqu'on étudie certaines structures algébriques, il est souvent intéressant de s'intéresser à des parties d'elles-mêmes ayant des propriétés remarquables et étudier comment elles interragissent (p.ex. $\mathbb{N}$ avec $\mathbb{Z}$ ou $\mathbb{R}$ avec $\mathbb{Q}$) et la façon de les construire. La conservation des propriétés de la structure est une préoccupation essentielle pour préserver la similarité entre une structure et ses parties.

À l'origine, le principe de sous-structure d'algèbre a été énoncé dans le cadre de la théorie des groupes de Galois dans la résolution exacte de polynômes [49] par l'application du groupe des permutations sur les racines. Cayley étant le premier à caractériser de manière abstraite la notion de groupe ce qui mènera au développement moderne de l'algèbre [2].

Ce concept a été ensuite étendu aux structures d'anneaux par Dedekind sous le nom d'idéal (*c.f.* par exemple [33]). La méthodologie a ensuite été généralisée et énoncée suivant des spécifications indépendantes de la structure et se révèlent importantes dans la formulation des théorèmes d'homomorphismes et d'isomorphismes. Ces derniers sont implicitement liés à la construction d'algèbre quotient dont l'usage est primordial en théorie des modèles algébriques.

2.2.2 Sous-structures et leurs propriétés

L'étude des parties stables d'une structure permet de présenter une dichotomie importante entre les relations et les opérations. En particulier, leur utilisation est incontournable dans la construction des représentations canoniques des structures algébriques et offre une perspective nouvelle sur l'étude des relations entre leurs éléments.

Une application célèbre étant la construction des cribles en théorie des nombres qui permettent d'identifier certaines de leur propriétés combinatoires. Ceux-ci exploite la

[2]Concernant l'addition, on devrait logiquement écrire $((a, b), c)) \in N^3$ afin de distinguer les opérandes du résultat dont la forme explicite est $= (+(a, b), c)$.

décomposition de tout nombre entier par la combinaison de facteurs afin d'éliminer tous les nombres qui ne peuvent être obtenus par combinaison, demeurent alors les nombres premiers.

Ces propriétés algébriques sont largement employées en cryptographie afin de concevoir des algorithmes de chiffrement des données réduisant les similarités entre deux messages chiffrés et entravant donc l'élaboration d'une méthode de déchiffrement fondée sur ce principe (*c.f.* par exemple [31] ou pour approfondir [94]). Inversement, ces techniques sont également impliquées dans la conception d'algorithmes gloutons pour le rendu de monnaie pour lequel on souhaite proposer un ensemble de pièces dont les valeurs respectives garantissent un nombre minimal de pièces rendues ([28, 89]).

On propose maintenant des définitions des parties remarquables d'une structure algébrique dans un contexte plus général que celui de l'arithmétique.

Définition 2.2.7 (Sous-relation)**.** *Soit deux structures de relations* $(A, (R_i))$ *et* $(B, (S_i))$ *de même type.* A *est une* sous-relation *pour* B *si et seulement si les conditions suivantes sont satisfaites :*

1. $A \subseteq B$
2. $(a_1, \ldots, a_n) \in R_i \Leftrightarrow a_1, \ldots, a_n \in A$ *et* $(a_1, \ldots, a_n) \in S_i$

L'interprétation est immédiate : tout élément de la partie préexiste dans le domaine original. Puis, la seconde condition impose que les tuples satisfaisant à chaque R_i dépende d'une sous-algèbre A et soient également valables sur B. Autrement dit, chaque relation R_i est la restriction de S_i par A. Dès lors, on voit facilement que la structure $(\mathbb{Z}, -)$ a pour sous-structure $(\mathbb{N}, -)$.

En effet, $\mathbb{N} \subsetneq \mathbb{Z}$ par construction, et toute opération valable dans $(\mathbb{N}, -)$ l'est dans $(\mathbb{Z}, -)$. Pourtant $(\mathbb{N}, -)$ n'est trivialement pas une algèbre par l'équation 2.2. Il faut donc imposer une définition de nature plus restrictive. Il est important de noter que la fermeture ne se fait pas suivant l'opération binaire de soustraction, $- : \mathbb{N} \times \mathbb{N} \to \mathbb{Z}$ mais par sa version unaire : $- : \mathbb{N} \to \mathbb{Z}^-$.

Selon la classification naturelle des structures algébriques, ni $(\mathbb{N}, -)$ ni $(\mathbb{Z}, -)$ ne présente généralement d'intérêt particulier, un magma étant la plus petite structure d'algèbre caractérisée par l'adjonction d'un opérateur binaire à un ensemble. Cependant, si ces deux structures sont dépourvues d'intérêt dans notre cas, elles se révèlent éclairantes pour illustrer le concept de fermeture.

Définition 2.2.8 (Sous-algèbre)**.** *Soit* $(A, (O_i))$ *et* $(B, (P_i))$ *deux structures algébriques similaires.* A *est une* sous-algèbre *de* B *(notée* $A \leq B$*) si et seulement si les conditions suivantes sont satisfaites :*

1. $A \subseteq B$
2. $a_1, \ldots, a_n \in A \implies O_i(a_1, \ldots, a_n) = P_i(a_1, \ldots, a_n)$

La seconde définition impose cette fois-ci que toute opération dans A retourne uniquement des éléments dans A. On en déduit que toute sous-algèbre est précisément une algèbre.

À l'inverse, cela implique que $(\mathbb{N}, -)$ avec l'opérateur unaire contient des éléments qui engendrent de nouveaux éléments n'appartenant pas à l'ensemble. Cela permet d'introduire le concept d'élément générateur, par opposition aux éléments distingués qui ne peuvent produire de nouvelles valeurs.

Lemme 2.2.1 (Intersection de sous-algèbres est une sous-algèbre)**.** *Soit (A_i) une collection non-vide de sous-algèbres de B indexée par un ensemble quelconque I*[3]*. Alors l'intersection des (A_i) est l'unique sous-algèbre $A = \bigcap_I A_i$.*

Démonstration. Sans perte de généralité, on présume que chaque algèbre est composée d'une unique opération binaire $O : B \to B$. Soit $A_i \leq B$, alors :

$$
\begin{aligned}
& a, b \in A && && (2.8)\\
\implies & (\forall i \in I)(a, b \in A_i) && (A \subseteq A_i) && (2.9)\\
\implies & (\forall i \in I)(O(a, b)) \in A_i && (A_i \text{ est fermée}) && (2.10)\\
\implies & O(a, b) \in A && (A_i \supseteq A) && (2.11)\\
\implies & A \leq B && && (2.12)
\end{aligned}
$$

□

La construction de la sous-algèbre dépend donc essentiellement de la vérification des propriétés ensemblistes sur le domaine sous-jacent à chaque algèbre. En particulier, l'intersection contient tous les éléments distingués de chaque opérateur de la structure et n'est donc jamais réduite à l'ensemble vide. Jusqu'à présent, nous avons identifiés la plupart des structures algébriques par l'ensemble des types d'opérateurs et de leurs éléments distingués. On distinguera par la suite les classes d'algèbres que l'on étudiera par les identités algébriques formant la base d'axiomes les caractérisant (p.ex. les groupes, les anneaux,...). La preuve précédente est énoncée en particulier dans ce cadre afin d'en faciliter la lecture.

Posons désormais l'ensemble E, une partie stricte d'une algèbre A. Si E n'est pas fermée, il se décompose en deux parties $E_1 \cup E_2 = E$, la première est celle des éléments distingués formant une sous-algèbre triviale, contenue dans toute sous-algèbre, la seconde est celle des générateurs telle que la fermeture soit une sous-algèbre de A.

Définition 2.2.9 (Sous-algèbre engendrée)**.** *Soit une algèbre $(A, \{O_i\})$ et une partie $E \subset A$, la sous-algèbre de A engendrée par E est la plus petite sous-algèbre C (au sens de l'inclusion) telle que :*

1. $E \subseteq C$

[3] Par la suite, sauf cas contraire, on considère que les indices sont à valeurs dans $\mathbb{N}$

2. $C \leq A$
3. $\forall B \leq A,\ E \subseteq B \Rightarrow C \subseteq B$

On note $A(E)$ la sous-algèbre engendrée par E. En pratique, la notation $\langle E \rangle$ sera préférée pour lever l'ambiguïté avec la structure ou son domaine sous-jacent.

Le lemme précédent (2.2.1) assure l'existence de $A(E)$ et on a le théorème suivant :

Théorème 2.2.1. *Soit la famille $\mathcal{A} = \{A_i\}$ des sous-algèbres de A. Soit une partie $E \subset A$. $A(E) = \bigcap_{A_i \in \mathcal{A}} \mathcal{A}$ est tel que $(\forall i)(E \subseteq A_i \leq A)$.*

Démonstration. Posons $A(E) \triangleq B = \bigcap_{A_i \in \mathcal{A}} \mathcal{A}$, on sait que $B \leq A$. On doit d'abord prouver que B est minimal. On pose $C \leq A$, on a :

$$E \subseteq C \qquad \text{(hyp.)} \tag{2.13}$$
$$\Longrightarrow C \in \mathcal{A} \qquad \text{(def.)} \tag{2.14}$$
$$\Longrightarrow B \subseteq C \tag{2.15}$$

Il reste à démontrer que B est nécessairement unique. Posons $B_1, B_2 \in \mathcal{A}$ telles que $B_1 \neq B_2$, on a :

$$|B_1| = |B_2| \text{(def.)} \tag{2.16}$$
$$\Longrightarrow \min(|B_1|, |B_2|) \geq |B_1 \cap B_2| \qquad \text{(hyp.)} \tag{2.17}$$
$$\Longrightarrow E \in B_1 \cap B_2 \tag{2.18}$$
$$\Longrightarrow B_1 \cap B_2 \leq A \tag{2.19}$$

Donc on en déduit que l'intersection de B_1 et B_2 est soit B_1 soit B_2, d'où $B_1 \subset B_2$ soit $B_2 \subset B_1$. Puisque $|B_1| = |B_2|$, on a ainsi $B_1 = B_2$, ce qui complète la preuve. □

On remarquera que la propriété d'unicité d'une sous-algèbre est due en totalité au fait que les structures algébriques sont avant tout des ensembles et que cette propriété s'applique donc à (presque) toutes les structures algébriques et permet donc de qualifier canoniquement certaines de leurs propriétés sans avoir à les reformuler.

Par exemple, la plus simple des structures algébriques est celle n'étant munie d'aucune opération. Par conséquent, il existe un ensemble par lequel tous les autres peuvent être construits, soit l'ensemble vide, noté $\{\}$ ou $\varnothing$. Considérant des ensembles munis d'une opération binaire tels $(\mathbb{N}, +, 0)$ et $(\mathbb{Z}, \times, 1)$, leurs plus petites sous-structures sont alors respectivement, $(\{0\}, +, 0)$ et $(\{1\}, \times, 1)$ et vérifient les équations triviales suivantes :

$$x + 0 = x \quad \forall x \in \{0\} \tag{2.20}$$
$$x \times 1 = x \quad \forall x \in \{1\} \tag{2.21}$$

Dans le cas présent, l'*oubli* de la structure, tel qu'il a été appliqué, n'induit pas d'effets de bord indésirables. En particulier, dans la majorité des cas qui seront évoqués dans ce manuscrit, l'abandon de la structure sera indolore et ne mènera pas à des constructions incohérentes.

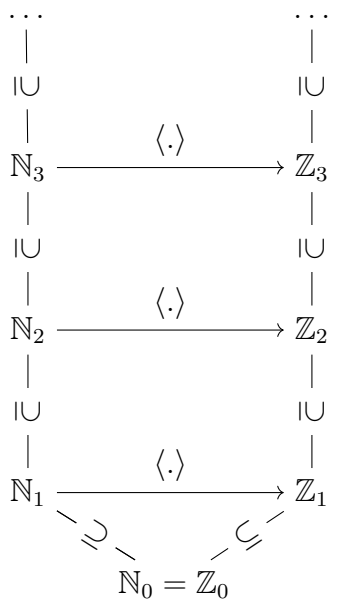


FIGURE 2.1 : Suite de générateurs $(\mathbb{N}, -)$ et leur sous-algèbre de $(\mathbb{Z}, -)$ ordonnées pour l'inclusion

Définition 2.2.10 (Générateur)**.** *Soit une algèbre $(A, \{O_i\})$ et une partie stricte X de A. On définit X comme étant un* ensemble générateur *de A ou X* engendre *A si et seulement si $\langle X \rangle = A$. Une algèbre A est* finiment engendrée *si elle possède un ensemble fini de générateurs.*

Cette définition implique que lorsque X engendre $(A, \{O_i\})$, tout élément de A peut être reproduit par une suite d'opérations finitaires sur des éléments de X. Pour certaines classes d'algèbres, cela permet de réduire celles-ci à des composantes spécifiques, exhibant les mêmes propriétés et d'améliorer leur compréhension. Reprenant l'exemple des tests de primalités évoqués en début de section, tout élément d'un groupe commutatif $x \in (G, +)$ et possédant un nombre fini de générateur, possède alors une représentation canonique sous la forme d'un vecteur d'éléments $(x_1, \ldots, x_k)$. Les générateurs x_i sont ainsi obtenues depuis des groupes cycliques $\mathbb{Z}/x_i\mathbb{Z}$ sous la condition qu'ils soient copremiers entre eux. Cela garantit qu'aucun d'entre eux ne puissent être décomposés à l'aide d'une combinaison des autres, assurant que la décomposition est unique.

Un autre exemple est celui de la représentation de toute algèbre de boole B comme des algèbres d'ensembles dont l'univers est la donnée des éléments générateurs de B, également appelés *atomes* (leurs propriétés seront largement discutées dans le prochain chapitre).

Représenter graphiquement la relation entretenue par la famille des sous-algèbres d'une algèbre A se révèle une tâche aisée. La figure (2.1) ordonne pour l'inclusion des sous-algèbres de $(\mathbb{Z}, -)$ par $\mathbb{Z}_n \triangleq [-n, n]$ engendrées sur des parties génératrices de $(\mathbb{N}, -)$ avec $\mathbb{N}_n \triangleq [0, n]$.

La flèche étiquetée par $\langle . \rangle$ illustre donc la fermeture algébrique suivante :

$$\langle . \rangle : (\mathbb{N}, -) \to (\mathbb{Z}, -) \tag{2.22}$$
$$\{x\} \mapsto \{x, -x\} \tag{2.23}$$

On voit naturellement que la sous-algèbre $\mathbb{Z}_0 = \{0\}$ est la plus petite partie fermée sur $\mathbb{Z}$ et est engendré par son élément distingué, soit $\langle \mathbb{N}_0 \rangle = \mathbb{Z}_0$.

On remarque également que chaque algèbre est *rangée* par taille et que la construction de chaque nouvel ensemble $\mathbb{N}_i \to \mathbb{N}_{i+1}$ découle de l'application de l'une des deux fonctions zero ou $\mathsf{succ}_{\mathbb{N}}\rangle$. En notant de manière analogue la construction des entiers relatifs, on obtient l'identité suivante :

$$\mathsf{succ}_{\mathbb{Z}} \circ \langle x \rangle = \langle \mathsf{succ}_{\mathbb{N}}(x) \rangle \tag{2.24}$$
$$\Leftrightarrow \ \mathsf{succ}_{\mathbb{Z}} \circ \langle . \rangle = \langle . \rangle \circ \mathsf{succ}_{\mathbb{N}} \tag{2.25}$$

pour tout $x \in \mathbb{N}_i$. On voit facilement à travers cette identité, à l'instar du diagramme, que l'application des fonctions $\langle . \rangle$ et $\mathsf{succ}_{\mathbb{Z}}$ commute. Il n'y a donc pas de restriction entre la construction itérative d'un ensemble d'entiers naturels et sa fermeture algébrique, y ajoutant les entiers relatifs correspondants. En outre, ce type de construction permet d'énoncé des propriétés *inductives* (relatives aux générateurs, donc) s'appliquant à l'ensemble des algèbres de même type.

De manière symétrique, en renversant l'ordre de lecture des flèches (à rebours de l'inclusion dans le diagramme), on peut considérer la décomposition suivante d'un ensemble par $\mathsf{succ}_{\mathbb{N}}(\mathbb{N}_i) = (i+1) + \mathsf{succ}_{\mathbb{N}}(\mathbb{N}_{i-1})$. On peut ensuite à loisir « déconstruire » canoniquement chaque ensemble avant ou après application de la fermeture et obtenir une identité de même nature mais sous un nouveau jour :

$$\langle \mathbb{N}_i \backslash \mathbb{N}_{i-1} \rangle = \langle \mathbb{N}_i \rangle \backslash \langle \mathbb{N}_{i-1} \rangle \tag{2.26}$$
$$\Leftrightarrow \langle \mathbb{N}_i \backslash \mathbb{N}_{i-1} \rangle = \mathbb{Z}_i \backslash \mathbb{Z}_{i-1} \tag{2.27}$$
$$\Leftrightarrow \langle \{i\} \rangle = \{i, -i, \ldots, 1, -1, 0\} \backslash \{i-1, -(i-1), \ldots, 1, -1, 0\} \tag{2.28}$$
$$\Leftrightarrow \langle \{i\} \rangle = \{i, -i\} \tag{2.29}$$

où l'opérateur de différence ensembliste $.\backslash.$ permet de distinguer une étape de fermeture entre deux étapes d'inductions.

On va ainsi s'intéresser au cas des fonctions préservant certaines propriétés au sein d'une famille de structures algébriques $\{A_i\}$ puis d'identifier des propriétés sur leur produit $\prod A_i$ et leur somme $\coprod A_i$. En effet, à l'image du diagramme (2.1) et la relation d'inclusion, les algèbres d'un même type admettent elles-mêmes un calcul mathématique d'ordre supérieur permettant de concevoir celles-ci en mettant l'accent sur les procédés de construction qui préservent la structure entre deux algèbres d'un même type ou transforment celles-ci en un autre type.

2.3 Homomorphismes et isomorphismes de structures

Le terme de morphisme renvoie à une sorte de fonction entre deux ensembles et vérifiant des propriétés intrinsèques aux ensembles de départ et d'arrivée. En particulier, leur étude porte principalement sur celles qui préservent la structure de l'ensemble de départ vers celui d'arrivée et donc associées à des structures concrètes. Dans ce cas, le terme plus général d'*homomorphisme* s'impose.

Par ailleurs, il est plus commode de différencier la nature de chaque morphisme $h : A \to B$ en examinant la relation engendrée sur $A \times B$ et les propriétés du graphe associé plutôt que sur des considérations purement algébriques et par conséquent moins intuitives au premier abord.

2.3.1 Généralités et Principes

Définition 2.3.1 (Homomorphisme de relations)**.** *Soit $(A, \{R_i\})$ et $(B, \{S_i\})$, deux structures de relations de même type. Une fonction $h : A \to B$ est appelée un homomorphisme $h : (A, \{R_i\}) \to (B, \{S_i\})$ de $(A, \{R_i\})$ vers $(B, \{S_i\})$ où :*

$$(a_1, \ldots, a_n) \in R_i \implies (h(a_1), \ldots, h(a_n)) \in S_i \tag{2.30}$$

$\forall i \in I$ *et* $\forall a_1, \ldots, a_n \in A$.

Cette définition énonce que la structure de départ est transférée à l'ensemble d'arrivée. De plus, à l'instar des fonctions opérant sur des ensembles, les morphismes peuvent ne pas être définis partout sur l'ensemble de départ ou d'arrivée. On a donc besoin d'une définition additionnelle garantissant une correspondance univoque soit entre A et B, soit entre B et A. Ceci permet de préciser la nature du morphisme et d'acquérir par la suite de nouvelles propriétés intéressantes par la composition de ceux-ci.

On considère de nouveau $(\mathbb{N}, -)$ et $(\mathbb{Z}, -)$, tous deux de même type et restreints aux n premiers entiers. On observe facilement que la relation binaire R associée à $(-)$ est l'ensemble $\{(0, 0)\}$ puisque $\mathbb{N}$ ne contient aucun inverse autre que trivial, tandis que la relation S associée à $\mathbb{Z}$ sera définie ainsi :

$$S \triangleq \{(a, b) \mid a + b = 0\} = \{(a, -a) \mid a \in \mathbb{Z}\} \tag{2.31}$$

$$\Leftrightarrow \{(a, -a) \mid a \in \mathbb{N}\} \cup \{(-a, a) \mid a \in \mathbb{N}\} \tag{2.32}$$

Un homomorphisme devrait garantir selon cette définition un minimum de *fidélité* entre les deux structures bien qu'il n'impose pas de restriction à la seconde. En effet, $(\mathbb{N}, -)$ est bien compatible avec $(\mathbb{Z}, -)$ mais ne collabore jamais à la structuration de ce dernier et le morphisme possède un caractère trivial. La notion de fidélité requiert alors pour chaque tuple dans S, l'existence d'antécédents qui soient en relation dans R.

L'opérateur de fermeture $\langle . \rangle : \mathbb{N} \to \mathbb{Z}$ n'est donc pas un homomorphisme sous ce prisme puisque par définition, il confère une structure à un ensemble qui n'en contient

pas ou partiellement. On verra plus loin qu'il est cependant composable avec des homomorphismes.

Définition 2.3.2 (Homomorphisme de relations)**.** *Soit (A, R) et (B, S) deux structures d'une seule relation binaire. Sans perte de généralité, une fonction $h : A \to B$ est appelée un homomorphisme $h : (A, R) \to (B, S)$ de (A, R) vers (B, S) où :*

$$(h(a), h(b)) \in S \implies \exists c \in A, (a, c) \in R,\ h(b) = h(c) \tag{2.33}$$

La définition algébrique induite est donc la suivante :

Définition 2.3.3 (Homomorphisme d'algèbres)**.** *Soit $(A, \{O_i\})$ et $(B, \{P_i\})$, deux structures algébriques, la fonction $h : A \to B$ est un homomorphisme si et seulement si :*

$$h(O_i(a_1, \ldots, a_n)) = P_i(h(a_1, \ldots, a_n)) \tag{2.34}$$

pour tout $i \in I$ et tout $a_1, \ldots, a_n \in A$

On vérifie aisément que la fonction $\mathbb{Z}_i \to \mathbb{Z}_{i+1}$ est un homomorphisme d'algèbre. En notant R et S, leur relation binaire respective comme énoncée en (2.31), on a pour tout $a, b \in \mathbb{Z}_i$:

$$\begin{cases} (a, b) \in S \implies \exists c, (a, c) \in R, b = c & (a \leq i \text{ et } b \leq i) \\ (a, b) \in S \text{ et } (a, b) \notin R \implies a, b \notin \mathbb{Z}_i & (a > i \text{ ou } b > i) \end{cases} \tag{2.35}$$

puisque $S = R \cup \{(i, -i), (-i, i)\}$. Donc sur l'intervalle $[0, i]$, on a le morphisme identité $h(x) = x$ et naturellement, les couples résultants ne peuvent être crées depuis $\mathbb{Z}_i$ et ne contredisent donc pas la définition (2.3.2).

On va maintenant énoncer quelques propriétés attachées aux homomorphismes.

Lemme 2.3.1. *Soit deux algèbres $(A, \{O_i\})$ et $(B, \{P_i\})$, on pose deux homomorphismes $h_1, h_2 : A \to B$, alors l'ensemble $E = \{x \in A \mid h_1(x) = h_2(x)\}$ est une sous-algèbre de A.*

Démonstration. Sans perte de généralité, chaque opérateur indexé est n-aire. On a $E \subseteq A$, donc pour tout $x_1, \ldots, x_n \in E$ et pour tout $i \in I$,

$$\begin{aligned} h_1(O_i(x_1, \ldots, x_n)) &= P_i(h_1(x_1), \ldots, h_1(x_n)) && \text{(def. 2.3.3)} && (2.36) \\ &= P_i(h_2(x_1, \ldots, h_2(x_n)) && \text{(def.)} && (2.37) \\ &= h_2(O_i(x_1, \ldots, x_n)) && \text{(def. 2.3.3)} && (2.38) \end{aligned}$$

□

Ce lemme garantit qu'il existe toujours une partie de A sur laquelle deux morphismes sont d'accord. Or, puisque $E \leq A$, il est intéressant de formuler E en fonction de A telle que les morphismes soient en accord sur tout A. La figure (2.2) illustre la présente définition où $f : E \to A$ est la fonction recherchée. Le théorème suivant fournit la solution.

$$E \xrightarrow{f} A \underset{h_2}{\overset{h_1}{\rightrightarrows}} B$$

FIGURE 2.2 : Diagramme de composition des morphismes $h_1 \circ f = h_2 \circ f$

Théorème 2.3.1. *Soit une algèbre A et une partie génératrice X de celle-ci. Soit $h_1, h_2 : A \to B$, on a, $\forall x \in X$:*

$$h_1(x) = h_2(x) \implies h_1 = h_2 \tag{2.39}$$

Démonstration. Comme précédemment, on pose $E = \{x \in X \mid h_1(x) = h_2(x)\}$ et on a :

$$\begin{aligned} \langle X \rangle = A & \qquad \text{(def.)} & (2.40)\\ \implies \langle X \rangle \subseteq E & \qquad \text{(def. 2.2.9)} & (2.41)\\ \implies E = A & & (2.42) \end{aligned}$$

La définition d'une sous-algèbre engendrée garantit que si deux morphismes sont en accord sur une partie génératrice X, ils le sont également sur sa fermeture $\langle X \rangle$. On en conclut donc que $h_1(x) = h_2(x), \forall x \in A$. □

2.3.2 Images, noyaux et compositions

Afin de mettre en relief le précédent théorème, on introduit la définition de l'image sous l'application d'une fonction.

Définition 2.3.4 (Image et Codomaine)**.** *Soit une fonction $f : A \to B$, on définit l'image de f par l'ensemble*

$$\mathrm{Im}(f) = \{b \in B \mid \exists a \in A, b = f(a)\} \tag{2.43}$$

aussi abrégé par $f[A]$. En notant $\mathrm{codom}(f) = B$, l'ensemble d'arrivée ou codomaine de f, alors $\mathrm{Im}(f) \subseteq \mathrm{codom}(f)$ est un corollaire trivial.

Lorsque l'image d'une fonction (ou d'un homomorphisme entre deux structures) est son ensemble d'arrivée, $\mathrm{Im}(f) = \mathrm{codom}(f)$, alors la fonction est une surjection et amènera à une relation importante entre algèbres.

Définition 2.3.5 (Composition de fonctions)**.** *Soit deux fonctions $f : A \to B$ et $g : B \to C$, $g \circ f : A \to C$ est l'unique fonction telle que $\mathrm{dom}(g) = \mathrm{codom}(f)$ avec :*

$$\{z \in C \mid \exists x \in A,\ \exists y \in B, f(x) = y \wedge g(y) = z\} \tag{2.44}$$

Lorsque la composition porte sur des homomorphismes de structures, on montre facilement que celle-ci est bien définie.

Théorème 2.3.2. *Soit deux homomorphismes $f : (A, \{O_i\}) \to (B, \{P_i\})$ et $g : (B, \{P_i\} \to (C, \{Q_i\})$, alors $g \circ f$ est un homomorphisme de A vers C.*

Démonstration. En constatant que tout homomorphisme commute avec les opérateurs de chaque structure, on a :

$$\begin{aligned}
(h \circ g)(O_i(a_1, \ldots, a_n)) &= h(g(O_i(a_1, \ldots, a_n))) && && (2.45)\\
&= h(P_i(g(a_1, \ldots, a_n))) && (g \circ O_i = P_i \circ g) && (2.46)\\
&= Q_i(h(g(a_1, \ldots, a_n))) && (h \circ P_i = Q_i \circ h) && (2.47)\\
&= Q_i((h \circ g)(a_1), \ldots, (h \circ g)(a_n)) && && (2.48)
\end{aligned}$$

□

On donne également une propriété importante sur l'image et la *préimage* d'un homomorphisme qui détermine les antécédents dans le domaine des images d'une fonction.

Théorème 2.3.3. *Soit $h : (A, \{O_i\}) \to (B, \{P_i\})$ un homomorphisme d'algèbres. On a la relation suivante entre les domaines de A et de B :*

$$h[X] \subseteq B \tag{2.49}$$
$$h^{-1}[X] \subseteq A \tag{2.50}$$

soit $h^{-1} \circ h[X] \subseteq A$, lorsque $X \subseteq A$.

Démonstration. On pose $a_1, \ldots, a_n \in X$, par la définition d'un homomorphisme (2.3.3), on a la relation suivante pour tout $i \in I$:

$$P_i(h(a_1, \ldots, a_n)) = h(O_i(a_1, \ldots, a_n)) \in h[X] \tag{2.51}$$

donc $h[X]$ atteint nécessairement une partie de B. Notons $Y \subseteq B$, la précédente équation prouve également que :

$$\begin{aligned}
h(a_1), \ldots, h(a_n) \in Y &\implies h(O_i(a_1, \ldots, a_n)) \in Y && (2.52)\\
&\implies O_i(a_1, \ldots, a_n) \in h^{-1}[Y] && (2.53)
\end{aligned}$$

et donc $h^{-1}[Y]$ est une partie de A. □

Une manière d'interpréter le théorème (2.3.1) est de considérer la relation binaire R induite par h_1 et h_2 sur $A \times B$ et suivant le diagramme (2.3). En effet, R est alors par construction un sous-ensemble du produit $A \times A$ et sur chaque composante de R, on vérifie que :

$$\begin{aligned}
(a, a') \in R \Leftrightarrow &\begin{cases} a \in \{x \in A \mid h_1(x) = h_2(x)\} \\ a' \in \{x \in A \mid h_1(x) = h_2(x)\} \end{cases} && (2.54)\\
\Leftrightarrow &\begin{cases} a \in h_1[A] \cap h_2[A] \\ a' \in h_1[A] \cap h_2[A] \end{cases} && (2.55)\\
\Leftrightarrow &(a, a') \in h_1[A] \cap h_2[A] \times h_1[A] \cap h_2[A] && (2.56)
\end{aligned}$$

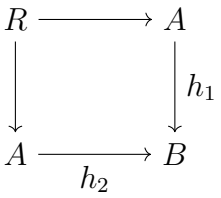


FIGURE 2.3 : Diagramme du produit $A \times A$

En utilisant la définition de l'image, on voit facilement que les couples de R sont tels que chaque composante est dans l'image commune aux deux morphismes. Lorsque h_1 et h_2 sont une seule et même fonction, on a donc la correspondance suivante :

$$(a, a') \in R \Leftrightarrow (a, a') \in S \tag{2.57}$$

où S est le produit sur l'image et par conséquent R est le produit sur le *noyau* dont on donne une définition générale.

Définition 2.3.6 (Noyau d'une fonction)**.** *Soit $f : A \to B$, on définit le* noyau *de la fonction f par :*

$$\ker(f) = \{(a, a') \in A \mid f(a) = f(a')\} \tag{2.58}$$

Généralement, lors de l'étude d'une structure concrète, on admet plutôt la définition ensembliste sur un sous-ensemble X de A tel que $\forall x, f(x) = 0$. Par rapport, à la définition générale, on repère dans ce cas les antécédents dans A qui ne sont pas inversibles depuis B puisque 0 désigne son élément distingué. Or toutes les algèbres ne sont pas nécessairement munies d'un élément de la sorte, donc la définition générale prévaut dans ce genre de situation. De plus, à l'instar de l'image, il est parfois souhaitable d'interpréter le noyau d'un morphisme comme un sous-ensemble de A plutôt que du produit $A \times A$ (illustré dans ce cas par la figure 2.4), dans ce cas, une définition alternative est possible :

$$\ker(f) = \{a \in A \mid \exists x, f(a) = f(x)\} \tag{2.59}$$

En règle générale, le premier intérêt du noyau d'un morphisme est la détermination de l'injectivité du morphisme : si deux éléments du domaine sont tels que $f(x) = f(y)$ alors ils seront nécessairement associés dans le noyau et par conséquent indifférentiables dans l'ensemble image. Donc un morphisme est injectif si son noyau est réduit à l'ensemble vide $\{\}$ ou à $\{0\}$ lorsque la structure contient un élément distingué.

Lorsque l'élément distingué existe dans le codomaine, modifier le diagramme précédent permet de construire aisément le noyau comme l'ensemble des images s'annulant en B comme le montre la figure (2.5).

Il suffit de voir que le noyau $\ker(f) \subset A \times A$ se décompose sous la forme de $A \times \{b\}$, pour chaque $b \in B$ tel que $f(a) = b$ et peut être reconstituée par la *réunion* de chaque partie telle que :

$$\ker(f) = \bigcup_{x \in \mathrm{Im}(f)} (A \times \{x\}) \tag{2.60}$$

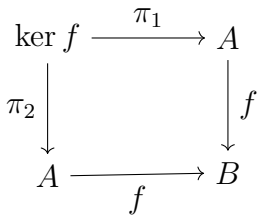


FIGURE 2.4 : Obtention du noyau de f selon le produit $A \times A$

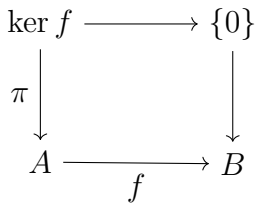


FIGURE 2.5 : Obtention du noyau de f selon le produit $A \times \{0\}$

Donc le noyau, lorsqu'il est vu comme une relation binaire, indexe les éléments du domaine obtenant la même image sous l'application du morphisme. Ce qui nous mènera à la définition d'espace quotient puis de relations d'équivalences dans la prochaine section.

Puisque le théorème 2.3.1 renseigne sur le fait que le noyau d'un homomorphisme $h : A \to B$ est nécessairement inclus dans le plus petit ensemble générateur X de A, on en déduit donc la propriété suivante :

Théorème 2.3.4. *soit $h : A \to B$, un homomorphisme et X une partie de A, on a pour tout $x \in X$:*

$$h \circ \langle x \rangle = \langle . \rangle \circ h(x) \tag{2.61}$$

On illustre facilement cette propriété en revenant à la construction de la famille d'algèbres $\{\mathbb{Z}_i\}$ dont une étape est représentée par le diagramme commutatif (2.6). Puisque $\mathbb{N}_i \subset \mathbb{N}_{i+1}$ et $\mathbb{Z}_i \subset \mathbb{Z}_{i+1}$, alors l'homomorphisme h est nécessairement l'injection canonique avec $h(x) = x$.

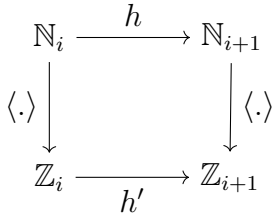


FIGURE 2.6 : Commutativité de l'homomorphisme et l'opérateur de fermeture : $\langle . \rangle \circ h = h \circ \langle . \rangle$

Comme pour le théorème (2.2.1), les concepts de noyaux et d'images sont donc souvent indépendants de la structure en considération et possède un caractère universel qui vont permettre d'énoncé des résultats fondamentaux pour la plupart des structures algébriques.

Définition 2.3.7 (Image homomorphe)**.** *Soit $h : A \to B$ un homomorphisme d'algèbres, B est l'*image homomorphe *de A si et seulement si h est un homomorphisme surjectif.*

Considérons cette fois-ci, le morphisme suivant :

$$h : (\mathbb{N}, +) \to (\mathbb{N}_1, +) \tag{2.62}$$
$$x \mapsto x \bmod 2. \tag{2.63}$$

qui identifie la parité d'un élément. On voit facilement que h est surjective puisque tout entier est nécessairement soit pair, soit impair. Cela n'établit cependant pas de correspondance fidèle entre les deux ensembles puisque h n'est pas injectif : on ne pourra plus différencier un entier pair d'un autre entier pair, et pareillement pour ceux impairs. Le noyau mesure donc les éléments indifférenciables et donc *équivalents* sous l'application de h.

Pour s'en convaincre, vérifions tout d'abord que le noyau de la fonction $h : x \mapsto x \bmod 2$, est une sous-algèbre de $(\mathbb{N}, +)$. Dans ce cas, le noyau construit l'ensemble des éléments pairs stables par composition, c.-à-d. assurant la fermeture de l'algèbre par opérations :

$$h(x) = 0 \tag{2.64}$$
$$h(x + y) = h(x) + h(y) = 0 \tag{2.65}$$
$$\tag{2.66}$$

On vérifie aisément que la partie engendrée est stable :

$$\ker(h) = \{0, 2, 4, 6, 8, \ldots\} \subset \mathbb{N} \tag{2.67}$$

donc $\ker(h) \leq \mathbb{N}$ et identifie les éléments pairs, soit les éléments dont le reste de la division par 2 vaut 0.

Si la correspondance est biunivoque entre A et B, c.-à-d. si h est une bijection, alors on a la propriété suivante sur les homomorphismes.

Définition 2.3.8 (Isomorphisme)**.** *Soit $f : A \to B$ un homomorphisme d'algèbres, f est un* isomorphisme *de A sur B si et seulement si il existe un morphisme $g : B \to A$ qui soit l'inverse à gauche et à droite de f soit : $(g \circ f)(x) = x \in A$ et $(f \circ g)(x) = x \in B$. Deux structures A et B sont isomorphes, $A \cong B$, si il existe un isomorphisme $A \to B$.*

On en déduit que la fonction f est à la fois une surjection puisque $\mathrm{Im}(h) = B$ et une injection puisque le noyau est trivial.

Un exemple récurrent d'isomorphisme est la correspondance entre les structures $(\mathbb{R}, +, 0)$ et $(\mathbb{R}, \times, 1)$ représentée par la fonction exponentielle et le logarithme qui sont des homomorphismes entre chaque structure :

$$\exp(x + y) = \exp(x) \times \exp(y) \tag{2.68}$$
$$\log(x \times y) = \log(x) + \log(y) \tag{2.69}$$

et vérifient les identités $(\exp \circ \log)(x) = (\log \circ \exp)(x) = x$.

Nous allons maintenant introduire les théorèmes fondamentaux permettant de construire des sous-algèbres particulières pour lesquelles les opérations de la structure initiale sont transportées.

2.4 Relations de congruence, algèbres quotients et modèles

On a vu dans la précédente section qu'il est parfois intéressant de construire des sous-structures à partir d'une algèbre de départ pour laquelle certaines propriétés sont préservées. On va maintenant s'intéresser à la nature de celles-ci et au procédé de construction permettant de les engendrer et sous quelles conditions.

Nous allons donc maintenant introduire des objets mathématiques utiles à la présente étude, on précise en particulier certaines propriétés propres à l'usage des relations.

Définition 2.4.1 (Composition entre relations)**.** *Soit un ensemble A muni ou non d'une structure d'algèbre. On pose une relation binaire R sur A telle que $R \subseteq A \times A$. Ainsi que dans la précédente sous-section, on préférera écrire $(a, b) \in R$ lorsque $a, b \in A$ sont en relation par A. On définit la composition des relations R et S comme la relation $R \circ S$:*

$$R \circ S \triangleq \{(a, b) \in A \times A \mid \exists c \in A, (a, c) \in R \wedge (c, b) \in S\} \tag{2.70}$$

On remarque que celle-ci est opportunément identique à celle de la composition des fonctions et morphismes où $R \circ S$ est identique à la composition des graphes de deux fonctions f et g définis toutes deux sur A.

Définition 2.4.2 (Relation d'équivalence sur A)**.** *Soit un ensemble A et une relation binaire R sur A. R est une* relation d'équivalence sur A *si les conditions suivantes sont vérifiées pour tout $a, b, c \in A$:*

$$(a, a) \in R \qquad \textit{(réflexivité)}$$
$$(a, b) \in R \implies (b, a) \in R \qquad \textit{(symétrie)}$$
$$(a, b) \in R \wedge (b, c) \in R \implies (a, c) \in R \qquad \textit{(transitivité)}$$

Dans ce cas, on préférera la notation $(a, b) \in \equiv$ ou celle plus naturelle $a \equiv b$. Nous admettrons également la notation $a \in [b]$.

Toute relation d'équivalence est naturellement associée à une partition d'ensemble sur le même ensemble.

Définition 2.4.3 (Partition définie sur A)**.** *Soit un ensemble A, une partition d'ensemble est définie comme une collection d'ensembles P telle que :*

$$\forall C, C' \in P, C \cap C' = \varnothing \tag{2.71}$$
$$\forall a \in A, \exists C \in P, a \in C \tag{2.72}$$

En particulier, la première propriété indique que les classes d'une partition sont *mutuellement exclusives* et la seconde, que celles-ci sont *conjointement exhaustives*.

Par exemple, le morphisme qui associe tout entier à sa parité dans $\{0, 1\}$ engendre donc deux collections formant une partition d'ensemble $P = \{\{0, 2, \ldots\}, \{\{1, 3, \ldots\}\}$, que l'on écrira également $024\ldots|135\ldots$ en l'absence d'ambiguïté.

Il est facile de mettre en évidence une relation d'équivalence canonique sur un ensemble A : la fonction identité $id : A \to A$ avec :

$$a \equiv b \Leftrightarrow id(a) = id(b) \Leftrightarrow a = b \tag{2.73}$$

La notion d'équivalence est donc rattachée ici avec l'égalité extensionnelle dans un ensemble : deux objets sont identiques si ils ont la même valeur ou extension dans cette ensemble. Dans ce cas, deux ensembles A et B sont égaux si ils contiennent exactement les mêmes objets.

L'axiome de séparation T_0 est la généralisation en topologie de ce concept ; lorsqu'on munit un ensemble A de la topologie discrète où chaque élément vu comme un point est isolé des autres. Autrement dit, aucun point n'est dans le voisinage d'un autre. On obtient alors la distance discrète associée :

$$d(a, b) = \begin{cases} 1 & a \neq b \\ 0 & a = b \end{cases} \tag{2.74}$$

où $d : A \times A \to E$ induit une relation d'équivalence des points par :

$$a \equiv_d b \Leftrightarrow d(a, b) = 0 \tag{2.75}$$

On a également vu dans la précédente section que les homomorphismes induisent des relations d'équivalences sur le noyau et l'image. Par exemple, posons $h : (A, +) \to (B, +)$, pour lequel il existe des éléments $a, b \in A$ tels que $h(a) = h(b)$, $+$ est un opérateur binaire. On a alors le fait suivant :

$$a \equiv_h b \Leftrightarrow h(a) = h(b) \tag{2.76}$$
$$\tag{2.77}$$

A et B étant des algèbres, la définition d'un homomorphisme requiert à cet égard que l'application de h commute avec l'application des multiples opérateurs ou relations de

la structure tel que $\forall x \in A, a + x \equiv_h b + x$. En particulier, lorsque $a \equiv_h b$ et $c \equiv_h d$, on a alors :

$$a + c \equiv_h b + d \tag{2.78}$$

qui admet la décomposition suivante :

$$a \equiv_h b \implies a + c \equiv_h b + c \tag{2.79}$$
$$c \equiv_h d \implies a + c \equiv_h b + d \tag{2.80}$$

Sans surprise, la relation d'égalité est une relation transitive. Ce qui nous amène à la définition de la relation de congruence.

Définition 2.4.4 (Relation de congruence)**.** *Soit une algèbre* $(A, \{O_i\})$ *et une relation d'équivalence* $\equiv$ *sur* A*.* $\equiv$ *est une* relation de congruence sur A *si les conditions suivantes sont vérifiées pour tout* i *:*

$$a_1 \equiv b_1 \wedge \ldots \wedge a_n \equiv b_n \implies O_i(a_1, \ldots, a_n) = O_i(b_1, \ldots, b_n) \tag{2.81}$$

Du fait qu'une relation d'équivalence implique nécessairement l'égalité dans l'image du morphisme, le remplacement d'un ou plusieurs termes congruents, tel qu'illustré plus haut, peut se faire pas à pas ou en une fois, sans conséquence.

On en déduit que toute fonction $A \to B$ est nécessairement associée à une relation d'équivalence et qu'un homomorphisme $A \to B$ l'est à une relation de congruence.

Cette définition est également l'analogue du principe de compositionalité de Frege. Interpréter les formules d'un langage (propositionnelle) revient alors à construire une algèbre $\mathcal{A}$ des formules sur la base des propriétés des connecteurs logiques – formant sa signature – et d'un ensemble de variables propositionnelles. Cette algèbre forme alors un modèle de calcul des propositions en fonction des spécifications syntaxiques définissant inductivement les formules. Dans le cadre axiomatique classique, si on considère une théorie par l'ensemble des formules Γ telle que $\forall \phi, \psi \in \Gamma$, on a alors l'assertion suivante :

$$\phi \equiv_\Gamma \psi \Leftrightarrow \Gamma \vdash \phi \leftrightarrow \psi \tag{2.82}$$

Lorsque l'interprétation d'une formule est une congruence, la signification globale d'une formule est alors entièrement déterminée par l'interprétation de ses composantes[4]. L'interprétation d'une formule et la valeur de celle-ci dans sa forme algébrique est libre de toute forme de contexte susceptible de la faire varier.

En particulier, l'évaluation est indépendante du contexte dans lequel est évalué une formule (isolée ou en tant que sous-formule). Plus précisément, chaque formule ne peut se voir attribuer qu'un seul modèle sémantique et donc une seule représentation algébrique.

[4] *cf.* également [16] pour une présentation pédagogique du lien étroit entretenu par la logique et l'algèbre

Système formel	Algèbre de Lindembaum-Tarski
Calcul propositionnel	Algèbre booléenne
Logique intuitionniste	Algèbre de Heyting
Logique des Prédicats	Algèbre cylindrique
	Algèbre polyadique

TABLE 2.1 : Relation entre divers systèmes formels $\mathcal{L}$ et leur algèbre quotient $\mathcal{L}/\equiv$

La combinaison de ces deux points caractérisent alors les *logiques extensionnelles* [54] pour lesquelles le modèle d'interprétation de chaque formule ne possède qu'une seule valeur sémantique (c.-à-d. valeur de vérité) et permet d'énoncé l'axiome d'extensionalité sur les formules logiques :

$$\forall \phi, \psi \in \mathcal{L}, [\![\phi]\!] = [\![\psi]\!] \rightarrow \phi = \psi \tag{2.83}$$

Celui-ci permet d'affirmer que si deux formules ne peuvent être distinguées par l'ensembles des propriétés algébriques les caractérisant, alors celles-ci sont égales.

Nous allons montrer maintenant que la classe d'équivalence sur A induit une structure quotient notée $A/_{\equiv}$; la relation induite étant souvent lue « A *modulo* $\equiv$ ».

Définition 2.4.5 (Structure quotient)**.** *Soit une algèbre $(A, \{O_i\})$ et une relation d'équivalence $\equiv$, si $\equiv$ est également une congruence alors on a l'identité suivante :*

$$P_i([a_1], \ldots, [a_n]) = [O_i(a_1, \ldots, a_n)] \tag{2.84}$$
$$\iff P_i \circ [.](a_1, \ldots, a_n) = [.] \circ O_i(a_1, \ldots, a_n) \tag{2.85}$$

où $\{P_i\}$ désignent les opérateurs dérivés.

Autrement dit, pour réaliser les opérations $\{P_i\}$ sur les classes d'équivalence dans A, il suffit de réaliser chaque $\{O_i\}$ sur un seul membre de chaque classe $[a_j]$. Pour facilier la compréhension, il suffit de voir que ce principe se réécrit $[a]+[b] = [a+b]$ dans $(\mathbb{N}, +)$.

Dans un contexte logique, $[\![\phi \text{ ou } \psi]\!] = [\![\phi]\!] \vee [\![\psi]\!]$ permet de déduire que la véracité de la disjonction entre un ensemble de propositions équivaut à évaluer la disjonction parmi les seules propositions atomiques dans ϕ et dans ψ.

Nous terminerons en démontrant que chaque homomorphisme détermine canoniquement une relation de congruence en utilisant la définition (2.3.7).

Théorème 2.4.1. *L'image homomorphe d'un morphisme $h : A \rightarrow B$ est isomorphe à une algèbre quotient sur A, et réciproquement.*

Démonstration. h détermine nécessairement une congruence puisque $a \equiv b \iff [a] = [b]$. Puisque h est surjectif, on a alors $h([a]) = [a]$, et puisque A est quotienté en chaque classe telle que $[a] = [b] \iff h(a) = h(b)$ alors $A/_{\equiv}$ est isomorphe à B. □

Lorsque nous aborderons les structures du treillis, nous aborderons quelque cas simples d'emploi de la construction quotient dans le but de rationaliser des ordres. Puis, nous présenterons dans le chapitre 3, un calcul algébrique sur les relations de congruence, émanant naturellement de requêtes d'agrégations.

3

Ensembles ordonnés, Treillis et leurs applications

Dans ce chapitre, nous allons présenter une classe de structures algébriques qui joue un rôle très important dans la modélisation algébrique de la logique.

En raison de la dualité existant entre structures relationnelles et algébriques, on aborde dans un premier temps le traitement des treillis sous la forme de relations et donc d'ensembles partiellement ordonnés.

3.1 Ensembles partiellement ordonnés

L'implication est un concept fondamental en logique. Celle-ci est le plus souvent interprétée comme une relation binaire opérant sur des phrases exprimées dans un langage ou sur des propositions. Suivant ce principe, le traitement des propositions dans une logique est exprimé dans un métalangage décrivant les règles admissibles de la déduction sous la forme d'une collection d'axiomes.

3.1.1 Préliminaires

Généralement, on modélise l'implication logique sous la forme d'une relation réflexive et transitive où les variables x, y, z sont quantifiées sur des phrases ou simplement sur

des propositions. Les relations vérifiant ces deux contraintes sont souvent nommées *préordre* ou *quasiordre*.

Définition 3.1.1 (Préordre)**.** *Soit un ensemble A et une relation binaire R sur A. R est un préordre si les conditions suivantes sont vérifiées pour tout $a, b, c \in A$.*

$$(a, a) \in R \tag{3.1}$$

$$(a, b) \in R, (b, c) \in R \implies (a, c) \in R \tag{3.2}$$

L'usage de ce nom vient du fait que ces relations sont « presque » des ordres puisque elles ne sont pas nécessairement symétriques ou antisymétriques. Selon le contexte, on définit symboliquement un préordre par $=$ ou $\lesssim$, le premier usage étant généralement réservé aux relations d'équivalences, le second est plutôt employé lors du processus de construction d'un ordre plus fort (complétion d'un ordre, relation de dominance, *etc*).

L'introduction de la propriété d'antisymétrie est plus délicate : énoncer la propriété par l'assertion $(a, b) \in R \implies (b, a) \notin R$ mène à invalider la propriété de réflexivité en empêchant la construction d'un ordre.

En effet, si on pose un ensemble A telle qu'il soit équipotent avec l'ensemble des entiers naturels, on a l'ordre suivant sur les éléments $a_0 \leq a_1 \leq a_2 \leq a_3 \leq \ldots$. Or, on ne dispose pas d'informations supplémentaires sur l'identité des éléments et qu'on admet un ensemble d'identités telles que $a_i = a_{i+1}$ pour tout $i \in \mathbb{N}$, alors (a_i, a_{i+1}) et (a_{i+1}, a_i) sont simultanément dans R, ce qui contredit la propriété.

Définition 3.1.2 (Antisymétrie)**.** *Soit un ensemble A muni d'une relation binaire R, R est* antisymétrique *si et seulement si la condition suivante est valide pour tout $a, b \in A$:*

$$(a, b) \in R \wedge (b, a) \in R \implies a = b \tag{3.3}$$

Cette définition appelle celle d'une relation d'ordre partiel.

Définition 3.1.3 (Poset)**.** *On appelle* ensemble partiellement ordonné *(ou* poset *pour* partially ordered set*, également* epo *pour ensemble partiellement ordonné) un ensemble P muni d'une relation d'ordre vérifiant les propriétés suivantes :*

1. $(\forall x \in P)(x \leq x)$ *(réflexivité)*
2. $(\forall x, y \in P)(x \leq y$ *et* $y \leq x \Rightarrow x = y)$ *(antisymétrie)*
3. $(\forall x, y, z \in P)(x \leq y$ *et* $y \leq z \Rightarrow x \leq z)$ *(transitivité)*

P est alors une relation d'ordre partiel.

Comme mentionné plus haut, la propriété d'antisymétrie s'obtient par le fait que la symétrie entraîne nécessairement l'identité sur les éléments impliqués dans l'ensemble. Si on matérialise une relation de préordre (A, R) à l'aide d'un graphe orienté, on est

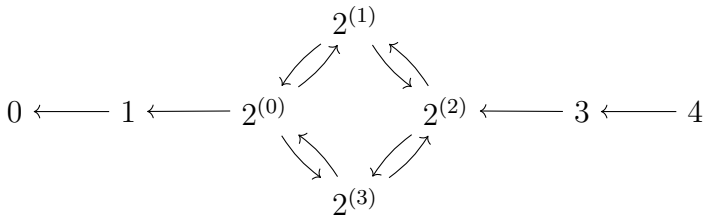


FIGURE 3.1 : Graphe de la relation d'ordre où $v_i \leftarrow v_j \implies v_i \leq v_j$

susceptible d'observer une bipartition de A en $A_{\leq} \cup A_{\lesssim}$ comme illustré par la figure (3.1).

Dans le présent cas, $A_{\lesssim} = \{2^{(i)} \mid \forall i\}$ et $A_{\leq} = \{0, 1, 3, 4\}$. On peut vérifier que ce dernier ensemble est tel que $\forall a, b \in A_{\leq}, (a, b) \in R \implies (b, a) \notin R$. Cela induit une nouvelle relation sur le produit $A_{\leq} \times A_{\lesssim}$.

Posons d'abord la fonction unaire f comme étant un analogue opérationel de R :

$$f : A \to \mathcal{P}(A) \tag{3.4}$$
$$a \mapsto \{b \in A \mid a \leq b\} \tag{3.5}$$

f associe donc tout sommet $a \in A$ à l'ensemble de ses successeurs dans le graphe. On peut dès lors vérifier la loi suivante :

$$f(a) \subseteq f(b) \wedge f(b) \not\subseteq f(a) \Leftrightarrow (a, b) \in R \wedge (b, a) \notin R \tag{3.6}$$
$$f(a) \subseteq f(b) \wedge f(b) \subseteq f(a) \Leftrightarrow (a, b) \in R \wedge (b, a) \in R \tag{3.7}$$

En observant que $E \subset F$ et $F \subset E$ équivaut à $E = F$, on obtient alors :

$$f(a) = f(b) \Leftrightarrow (a, b) \in A_{\lesssim} \tag{3.8}$$

qui se transpose sur l'ensemble de la relation par

$$\forall a, b \in A, (a, b) \in R \wedge (b, a) \in R \Leftrightarrow f(a) = f(b) \tag{3.9}$$

qui renvoie à la définition de l'antisymétrie.

La fonction f induit donc une relation d'équivalence sur A, avec $a \equiv_f b \Leftrightarrow (a, b) \in A_{\lesssim}$ et on obtient :

$$(a, b) \in R \Leftrightarrow a < b \vee a = b \tag{3.10}$$

La figure (3.2) représente la partition des sommets selon ce principe.

On peut dès lors quotienter $A \times A$ de sorte à obtenir une relation d'ordre partielle sur A par :

$$[a] \leq / \equiv [b] \Leftrightarrow a \leq b \tag{3.11}$$

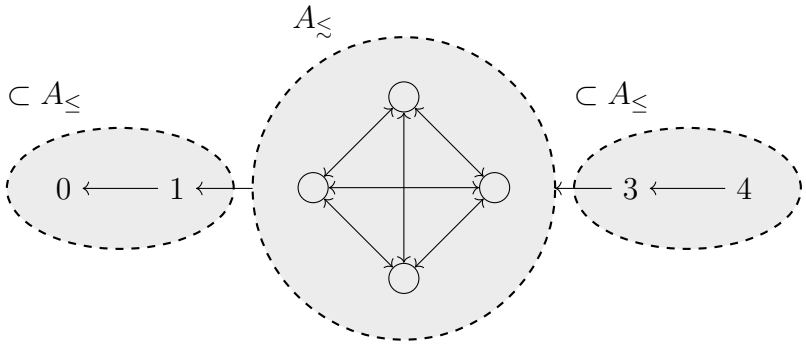


FIGURE 3.2 : Représentation de R par la bipartition $A_{\leq} \cup A_{\lesssim}$

$$[0] \longleftarrow [1] \longleftarrow [2] \longleftarrow [3] \longleftarrow [4]$$

FIGURE 3.3 : Graphe de la relation $\leq/\equiv$

où $[x] = \{a \mid f(a) = f(x)\}$

Dans notre cas, chaque élément $a \in A$ forme sa propre relation d'équivalence $[a] \in A \times A$, hormis les éléments dans $A_{\lesssim}$ tous arbitrairement représentés par la classe $[2] = \{2^0, 2^1, 2^2, 2^3\}$. Le graphe de la figure (3.3) matérialise la relation d'ordre partiel ainsi obtenue.

On constate que l'ordre ainsi obtenu ne vérifie plus la réflexivité par voie de conséquence. Cet aspect sera intéressant lorsqu'on évoquera la représentation graphique habituelle des ordres et la construction de relation de couverture par l'élimination d'éléments[1] *redondants* dans la relation $R \subseteq A \times A$. En somme, l'ensemble des couples réflexifs de la relation et qui ne participent pas à rendre identifiables les parties du domaine satisfaisant à la relation.

Définition 3.1.4 (Connexité)**.** *Soit un ensemble A muni d'une relation binaire R sur A. R est* connexe *sur A si et seulement si, on vérifie la condition suivante, pour tout $a, b \in A$:*

$$(a, b) \in R \vee (b, a) \in R \tag{3.12}$$

La relation $\leq/\equiv$ est donc connexe. En particulier, dans le cas où la relation est un ordre partiel, on obtient la définition d'une relation d'ordre total.

Définition 3.1.5 (Relation d'ordre total)**.** *Soit un ensemble A muni d'une relation binaire R sur A. R est une relation d'ordre total si et seulement si :*

1. *R est une relation d'ordre partiel*
2. *R est connexe*

[1] Les couples (x, y) d'une relation binaire R sont considérés dans ce cas comme des éléments ponctuels et remarquables, à l'instar des arêtes dans un graphe non-oriénté.

L'ordre sur les entiers naturels est à l'évidence une relation d'ordre total sur $\mathbb{N}$ qui garantit que pour tout couple $(a, b) \in \mathbb{N} \times \mathbb{N}$, on peut soit établir $a \leq b$ soit $b \leq a$ et $(\mathbb{N}, \leq)$ est un ensemble totalement ordonné.

Cependant, toutes les relations entretenues sur $\mathbb{N}$ ne sont pas nécessairement des ordres totaux. Le relation de divisibilité en est un exemple : b divise a, symboliquement $a|b$ par :

$$a|b \Leftrightarrow \exists k \in \mathbb{N}, a = k \times b \tag{3.13}$$

Il est facile de voir que cette relation est naturellement réflexive et antisymétrique. De plus, par l'associativité de la multiplication entière, la relation est transitive :

$$a|b \wedge b|c \Leftrightarrow \exists k, k', a = k \times (k' \times c) \tag{3.14}$$
$$\Leftrightarrow a = (kk') \times c \tag{3.15}$$
$$\Leftrightarrow a|c \tag{3.16}$$

L'ensemble des entiers naturels est alors un ensemble partiellement ordonné $(\mathbb{N}, .|.)$. En revanche, le caractère représentable de la relation est moins évident que le précédent exemple puisque fortement combinatoire, même pour un petit nombre d'éléments. L'utilisation de la précédente relation d'équivalence $\equiv_f$ va se révéler d'intérêt pour mettre en lumière la combinatoire induite par la relation et en améliorer son analyse.

Posons par exemple, $A = \{1, 2, 3, 5\}$, figurant l'ensemble des quatre premiers, nombres premiers. La fonction suivante trouve l'ensemble des successeurs pour la relation de divisibilité, de manière analogue à l'exemple précédent :

$$f : \mathbb{N} \to \mathcal{P}(A) \tag{3.17}$$
$$a \mapsto \{b \in A \mid \exists k \in \mathbb{N}, a = k \times b, k \neq 1\} \tag{3.18}$$

Premier constat : f engendre des éléments n'appartenant pas à A. On construit alors l'ensemble $A^+ \triangleq \{y \in \mathbb{N} \mid \forall x \in A, y \in f(x)\}$. $f(x)$ contient donc l'ensemble des facteurs $k_1, k_2, \ldots, k_n$ tels que

$$x = \prod_{k_i \in f(x)} k_i \tag{3.19}$$

On peut facilement vérifier que la relation peut s'exprimer dans un vocabulaire purement ensembliste :

$$f(a) \supset f(b) \Leftrightarrow a|b \tag{3.20}$$

Par exemple, on a $f(10) = \{2, 5\} \supset f(5) = \{5\}$, tandis que $10|5$ puisque $10 = 2 \times 5$.

On peut alors définir une classe d'équivalence sur les parties de A par :

$$[a] = \{x \mid f(x) = f(a)\} \tag{3.21}$$

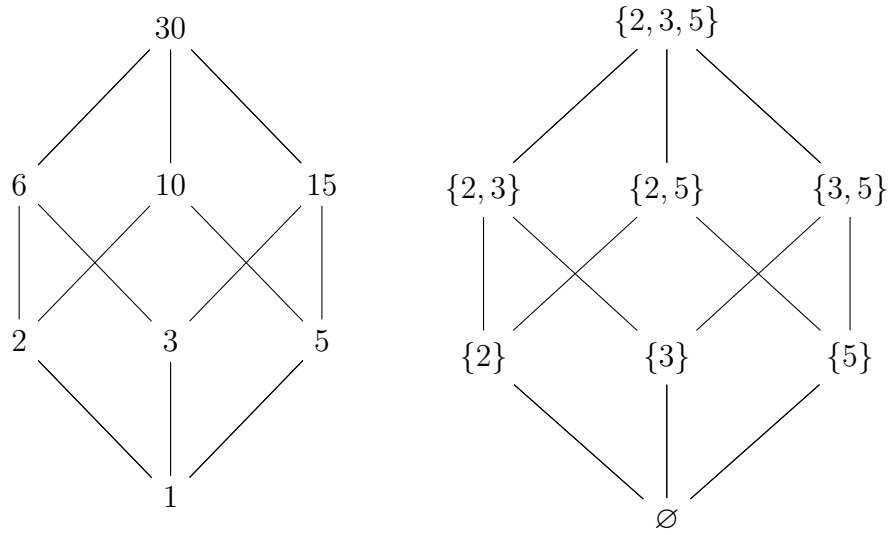


FIGURE 3.4 : Graphe de la relation de divisibilité $(A^+, .|.)$ *vs.* la relation $(\mathcal{P}(A), \subseteq)$

Celle-ci établit une dichotomie entre les nombres premiers dont les classes ne contiennent que le nombre lui-même, $[a] = \{a\}$, et l'ensemble des entiers obtenus par produit de nombres premiers. La figure (3.4) représente une telle configuration où chaque arête figure l'inclusion ensembliste entre les classes d'équivalences.

De manière prévisible, chaque ensemble dans le graphe de droite représente une unique décomposition en facteurs premiers. Néanmoins, les classes d'équivalences ne sont pas triviales car elles ne sont pas réduites au singleton. En effet, si on considère la définition explicite des classes par :

$$[x] \triangleq \{ \{k_i\} \subseteq A \mid \forall_i, \exists \alpha_i, x = \prod k_i^{\alpha_i}, |[k_i]| = 1, \alpha_i >= 1\} \tag{3.22}$$

celle-ci renvoie au théorème fondamental de l'arithmétique et donne une définition canonique de tout entier sous la forme d'un produit de nombres premiers.

La représentation ensembliste élimine donc toutes les représentations sous forme de facteurs impliquant des coefficients α_i différents de 1 et on a par exemple $[2] = [4] = [8] = \{2\}$ ou $[3] = [9] = \{3\}$. En notant, $A = f(a)$ et $B = f(b)$, on peut illustrer ce phénomène par le biais de l'identité ensembliste suivante :

$$A \subset B \Leftrightarrow B = A \cup B \tag{3.23}$$

$$\Longrightarrow A \subset B \wedge B \subset A \Leftrightarrow A = B \tag{3.24}$$

qui implique que tout ensemble est distingué[2] et $A = A \cup A$. Or, le seul élément distingué pour la multiplication entière est 1 et donc $\forall a \in \mathbb{N}, a \times a = a^2 \neq a$.

La figure (3.5) différencie la classe d'équivalence de $[2]$ et de $[4]$ et construit donc de nouveaux nombres se décomposant en 2^2. Cela engendre par la suite un nouvel ordre partiel plus riche que le précédent comme le montre la figure (3.6).

[2]Dans le sens utilisé jusqu'à présent : lorsqu'il est combiné avec lui-même, il est son propre résultat.

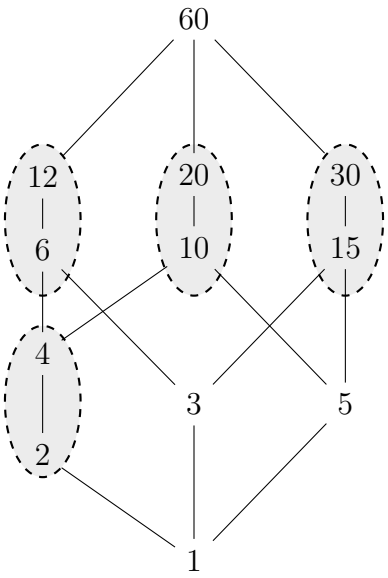


FIGURE 3.5 : Graphe de la relation de divisibilité $(A^+, .|.)$ et les nombres $n = 2^2 \times k$.

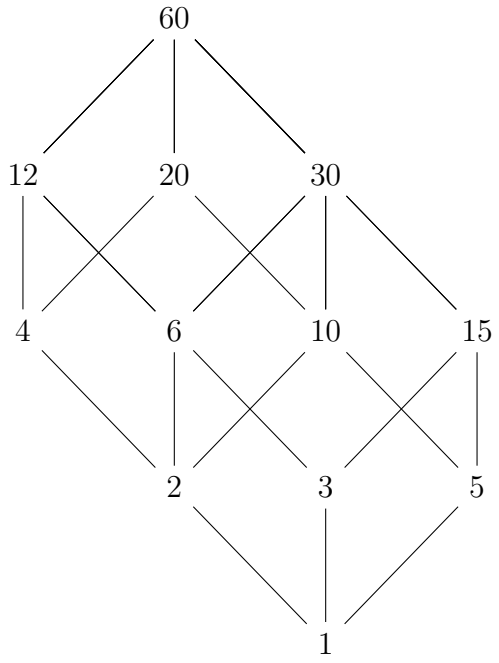


FIGURE 3.6 : Graphe de la relation de divisibilité $(A^+ \times 2, .|.)$

La propriété d'antisymétrie ne peut donc pas se généraliser à tous les cas de figure. En particulier, toutes les structures d'ordre partiel ne sont pas nécessairement représentables sous forme d'ensembles munis de la relation d'inclusion.

3.1.2 Liens avec les ordres stricts

On va considérer ici quelques généralisations au concept d'ordre partiel aux relations généralement transitives, soit des *ordres* et incluant les spécialisations que l'on vient d'évoquer, soit les préordres, les ordres partiels et les ordres totaux.

On considère par la suite des relations irréflexives permettant de définir des ordres stricts.

Définition 3.1.6 (Relation d'ordre strict)**.** *Soit un ensemble A muni d'une relation binaire R. R est un* ordre strict *sur A si les conditions suivantes sont vérifiées :*

1. *R est irréflexive sur A*
2. *R est transitive sur A*

De manière analogue, une structure (A, R) est alors un *ensemble strictement ordonné* si R est un ordre strict sur A. Comme on l'a vu avec les deux exemples précédents, toute structure d'ordre partiel R admet une présentation sous la forme d'un ordre strict R^* par l'élimination des éléments réflexifs. Lorsque la transitivité est présente, irréflexivité et asymétrie coïncident comme le montre la figure (3.3).

On propose maintenant le lemme suivant établissant la propriété que l'on avait admis précédemment sans la démontrer.

Lemme 3.1.1. *Soit un ensemble A muni d'une relation binaire R telle que R soit transitive. On a la propriété suivante :*

$$R \text{ asymétrique} \Leftrightarrow R \text{ irréflexive}$$

Démonstration. ($\Leftarrow$) Prouvons d'abord que si R est irréflexive, R est nécessairement asymétrique. Pour tout $a \in A$, on a :

$$(a, a) \notin R \qquad \text{(def.)} \qquad (3.25)$$
$$(a, b) \in R \wedge (b, c) \in R \implies (a, c) \in R \qquad \text{(def.)} \qquad (3.26)$$
$$(3.27)$$

On suppose que la propriété est fausse, alors on a :

$$(a, b) \in R \implies (b, a) \in R \qquad \text{(hyp.)} \qquad (3.28)$$
$$\implies (a, a) \in R \qquad \text{(def.)} \qquad (3.29)$$
$$(3.30)$$

Par transitivité, cela mène donc à une contradiction. Donc R est nécéssairement asymétrique. ($\Rightarrow$) Soit R asymétrique, alors on a :

$$(a, b) \in R \implies (b, a) \notin R \tag{3.31}$$

On suppose que R est réflexive, alors on a :

$$(a, a) \in R \implies (a, a) \notin R \tag{3.32}$$

par définition de la symétrie et c'est donc une contradiction. Donc R est nécessairement irréflexive. On déduit qu'en présence de la transitivité, irréflexivité et asymétrie coïncident, ce qui conclut la preuve. □

Cette démonstration permet de déduire que tout ordre partiel peut être associé à un ordre strict et réciproquement, et l'un et l'autre sont donc plus ou moins des concepts équivalents.

Le théorème suit naturellement.

Théorème 3.1.1. *Soit une structure (A, R), R est un ordre strict et les deux définitions suivantes sont équivalentes :*

1. *R est transitive et antisymétrique*
2. *R est transitive et irréflexive*

Dans la même veine, on peut définir par analogie, le concept d'ordre linéaire strict. Pour cela, on met en avant une restriction de la propriété de connexité.

Définition 3.1.7 (Relation faiblement connexe)**.** *Soit un ensemble A muni d'une relation binaire R. R est* faiblement connexe *si la condition suivante est satisfaite pour tout $a, b \in A$:*

$$(a, b) \in R \text{ ou } (b, a) \in R \implies a \neq b \tag{3.33}$$

qui se lit également par la propriété de trichotomie *: soit $(a, b) \in R$, soit $(b, a) \in R$, soit $a = b$.*

Définition 3.1.8 (Relation d'ordre linéaire strict)**.** *Soit un ensemble A muni d'une relation binaire R. R est un ordre linéaire strict si et seulement si les conditions suivantes sont satisfaites :*

1. *R est transitive*
2. *R est irréflexive*
3. *R est faiblement connexe*

(A, R) est alors un ensemble totalement ordonné strict.

Un principe de dualité autre que celui connectant ordres stricts et ordres partiels peut être considéré. Définissons tout d'abord l'inverse d'une relation R.

Définition 3.1.9. *Soit un ensemble A muni d'une relation R. L'inverse de R est défini par la formule suivante :*

$$R^{-1} \triangleq \{(x, y) \mid (y, x) \in R\} \tag{3.34}$$

L'inversion permet de considérer une structure de relation ordonnée duale de R partageant des propriétés communes.

Théorème 3.1.2. *Soit deux structures (A, R) et (A, R^{-1}), telles que R^{-1} est l'inverse de R. Si R est un ordre, alors R^{-1} est également un ordre.*

Démonstration. Soit R une relation transitive, alors on a :

$$(a, b) \in R \wedge (b, c) \in R \implies (a, c) \in R \qquad \text{(def.)} \tag{3.35}$$
$$(b, a) \in R^{-1} \wedge (c, b) \in R^{-1} \implies (c, a) \in R^{-1} \qquad \text{(def.)} \tag{3.36}$$

Donc R^{-1} est également transitive. □

On peut également prouver que R^{-1} hérite de la plupart des propriétés énoncées jusqu'à maintenant dans cette section.

3.1.3 Relation de couverture et Diagramme de Hasse

Comme les représentations graphiques introduites précédemment l'ont suggéré, les relations d'ordre partiel admettent des illustrations naturelles rendant compte de la nature de celles-ci. Les représentations associées sont appelées *Diagramme de Hasse* et sont basées sur la notion de couverture d'une relation.

Avant d'aller plus loin, il est nécessaire d'introduire quelques définitions qui vont permettre de faire la transition avec ce qui vient d'être établi sur les relations d'ordre partiel.

Définition 3.1.10 (Fermeture transitive)**.** *Soit R une relation binaire sur un ensemble A. On appelle* fermeture transitive *de R, symboliquement R^+, la plus* petite relation transitive *dans R. Plus précisément, on a l'équivalence suivante :*

$$(a, b) \in R^+ \Leftrightarrow \begin{cases} (a, b) \in R \text{ ou} \\ \exists x_1, \ldots, x_n \in A, (a, x_1) \in R \wedge \ldots \wedge (x_n, b) \in R \end{cases} \tag{3.37}$$

Si on se remémore la dualité entre structures algébriques et structures de relations, la notion est semblable à celle de la fermeture d'une sous-algèbre munie d'un opérateur unaire (*c.f.* 2.2.8 et 2.2.9). Le calcul de la plus petite relation transitive est alors similaire à celui de la plus petite sous-algèbre engendrée par une famille d'algèbres (*c.f.* 2.2.1).

Lemme 3.1.2. *Soit R une relation binaire sur A. On pose la famille de relations transitives $\{R_i\}$, alors $\bigcap_i R_i$ est également transitive.*

Démonstration. Posons R_i, R_j toutes deux transitives avec $(a,b),(b,c) \in R_i \cap R_j$ pour tout i et j, alors on a :

$$\begin{aligned} & (a,b) \in R_i, (a,b) \in R_j && \text{(def. } \cap\text{)} && (3.38)\\ \wedge & (b,c) \in R_i, (b,c) \in R_j && \text{(def. } \cap\text{)} && (3.39)\\ \Longrightarrow & (a,c) \in R_i, (a,c) \in R_j && \text{(def.)} && (3.40)\\ \Longrightarrow & (a,c) \in R_i \cap R_j && \text{(def. } \cap\text{)} && (3.41) \end{aligned}$$

On en déduit que $R_i \cap R_j$ est transitive. Donc $\bigcap_i R_i$ l'est également, ce qui complète la preuve. □

En particulier, on peut démontrer que la dimension du plus petit ensemble générateur d'une relation binaire est de dimension 3. Cela signifie que la cardinalité d'une plus petite sous-algèbre est bornée par le nombre d'éléments nécessaires pour former les relations induites pour chaque symbole d'opérations inclus dans la signature.

Le résultat suivant est une variante de celui concernant la sous-algèbre engendrée.

Théorème 3.1.3. *Soit R une relation binaire sur A. L'intersection de la famille des relations $\{R_i\}$ et contenant R, est la plus petite relation transitive contenant R.*

Démonstration. Il est facile de démontrer qu'il existe au moins une relation binaire sur A qui contient R : la relation triviale $A \times A$ est par définition transitive et contient donc R.

On pose $S = \bigcap R_i$ telle que $R \subseteq R_i$ pour tout i. Le lemme précédent dit que S est transitive donc on a :

$$R \subseteq S \tag{3.42}$$

puisque l'intersection préserve les sous-ensembles. Il existe alors une relation transitive R' qui contient R et on doit avoir $S \subseteq R'$. On en déduit que S est minimal et donc la fermeture transitive $R^+ = S$. □

On définit de manière analogue la fermeture réflexive et transitive.

Définition 3.1.11. *Soit R une relation binaire sur un ensemble A. On note R^* la fermeture réflexive et transitive de R par la plus petite relation contenant R et vérifiant l'une ou l'autre des conditions suivantes, pour tout $(x,y) \in R^*$:*

1. *soit $x = y$*
2. *soit $(x,y) \in R^+$*

Puisque la fermeture transitive R^* existe pour toute relation R, on peut toujours construire le préordre également associé à une relation. De plus, toute relation peut être associée naturellement à un (pré)ordre. Il reste à déterminer sous quelles conditions ce dernier est un ordre partiel, strict ou non.

Définition 3.1.12 (Relation bien fondée)**.** *Soit R une relation binaire sur un ensemble A. R est* bien fondée *si et seulement si sa fermeture transitive R^* est* bien fondée*. Ce qui implique de vérifier la séquence (possiblement infinie) suivante, soit $\forall n \in \mathbb{N}$:*

$$(x_0, x_1) \in R^* \implies x_0 \neq x_1 \tag{3.43}$$

$$\vdots$$

$$(x_0, x_1) \in R^* \wedge \ldots \wedge (x_{n-1}, x_n) \in R^* \implies x_0 \neq x_n \tag{3.44}$$

Une définition alternative est souvent donnée par l'*axiome de régularité* de la théorie des ensembles qui énonce qu'il existe pour toute partie $S \subseteq R$, un élément x qui ne soit pas en relation avec tous les autres tel que $(x, s) \notin S$. La définition précédente entend qu'il existe une numérotation des objets dans l'univers (l'ensemble est équipotent avec $\mathbb{N}$) telle que cet ordre soit compatible avec l'ordre induit par la relation. Il est alors possible de démontrer la validité d'une propriété (c.-à-d. relation) par induction sur les éléments de A.

Lorsqu'on évoquera les propriétés combinatoires associées aux ordres partiels dans la prochaine sous-section, on parlera dans ce cas de minimalité ou de maximalité (en vertu de la dualité connectant un ordre à son inverse), propriété indispensable pour définir un calcul de point-fixe et donc un calcul algébrique dans les treillis.

Il est assez facile de trouver une illustration de cette définition en présentant un contre-exemple algébrique. Posons la structure algébrique $(\mathbb{Z}, +)$ et le morphisme d'algèbre suivant :

$$h_a : \mathbb{Z} \to \mathbb{Z} \tag{3.45}$$

$$x \mapsto x \times a \tag{3.46}$$

avec a un entier positif non nul. L'image du morphisme est alors une partie fermée de $\mathbb{Z}$ (une sous-algèbre) représentant l'ensemble des multiples de a. Le noyau de h_a définit donc la classe d'équivalence suivante :

$$\ker(h_a) = \{x \in \mathbb{Z} \mid a \times x = 0\} \tag{3.47}$$

Par définition, il existe une paire (x, x') pour laquelle $h_a(x) = h_a(x')$, l'addition ayant un inverse dans $\mathbb{Z}$, cela implique $h_a(x) - h_a(x') = 0$ et par définition d'un homomorphisme, $h_a(x - x') = 0$ et finalement $x - x' \in \ker(h_a)$. Il existe alors un entier n tel que :

$$x = an \times x' \tag{3.48}$$

$$\Leftrightarrow x = x' \bmod n \tag{3.49}$$

Par le premier théorème d'isomorphisme, le quotient $\mathbb{Z}/\ker h$ induit une sous-algèbre fermée, soit $\mathbb{Z}/n\mathbb{Z}$ dont l'opération associée devient l'addition *modulo* n. En notant h' l'homomorphisme inverse tel que $h' \circ h = h \circ h'$:

$$h' : \mathbb{Z} \to \mathbb{Z}/n\mathbb{Z} \tag{3.50}$$

$$x \mapsto x \bmod n \tag{3.51}$$

on observe facilement que pour tout couple d'entiers (x, y) l'ordre sur les entiers n'est pas préservé dans $\mathbb{Z}/n\mathbb{Z}$ tel que $x \leq y \implies h'(x) \leq h'(y)$. En effet, cette structure étant un groupe cyclique, il existe alors un entier suffisamment grand tel que l'opération s'annule.

En notant R la relation binaire associée au morphisme h' par $(x, y) \in R \Leftrightarrow h'(x) < h'(y)$, on a la séquence suivante :

$$(x_0, x_1) \in R \wedge \ldots \wedge (x_{n-1}, x_n) \in R \implies h(x_n) = h(x_0) \tag{3.52}$$

qui s'illustre facilement avec $n = 2$ et en observant que pour un couple d'entiers $(x, x + 2)$:

$$x < x + 2 \implies h'(x) = h'(x + 2) \tag{3.53}$$

La relation induite n'est donc pas un ordre bien fondé.

Une application simple des ordres bien fondés est la preuve de la terminaison d'un programme récursif ou de l'arrêt d'une boucle en un nombre fini d'étapes.

Un autre exemple de relation transitive pouvant être cyclique est le résultat d'une élection obtenue par la procédure de vote de Condorcet. On présume que des individus sont représentés par des éléments dans $\mathbb{N}$ dont on demande de réaliser un classement par ordre de préférences parmi trois propositions $A = \{a, b, c\}$. On obtient alors une famille d'ordres stricts $\{>_i\}$ où chaque individu est en mesure de désigner sans ambiguité (ou de manière rationnelle) une préférence pour chaque couple de propositions.

Le tableau suivant présente un tel choix pour trois individus :

Votant	Préférence n°1	Préférence n°2	Préférence n°3
i_1	a	b	c
i_2	b	c	a
i_3	c	a	b

TABLE 3.1 : Une collection de préférences $\{>_1, >_2, >_3\}$

On suppose ensuite qu'un choix est légitime si il est entériné par la majorité des individus : x est préféré à y, soit $x > y$ si et seulement si $\exists i, j$ tels que $x >_i y$ et $x >_j y$ ou de manière simplifiée $x >_{i,j} y$. On lit alors depuis la table les faits suivants : $a >_{1,3} b$, $b >_{1,2} c$ et $c >_{2,3} a$.

L'ordre obtenu $>= \{(a,b),(b,c),(c,a)\}$ est alors l'union des préférences majoritaires et n'est donc pas un ordre bien fondé. De manière plus générale, lorsque $(A,>)$ est défini par $\bigcup\bigcap_i >_i$, on peut démontrer que l'union de plusieurs ordres bien fondés $\{>_i\}$ n'est pas nécessairement un ordre bien fondé.

Théorème 3.1.4. *Soit R une relation bien fondée. Alors R^* est un ordre strict et R^+ est un ordre partiel.*

Définition 3.1.13 (Relation de couverture)**.** *Soit un ensemble partiellement ordonné $(A,\leq)$, soient deux éléments $a,b \in A$, b* couvre *a si et seulement si les conditions suivantes sont satisfaites :*

1. $a < b$
2. $\nexists x \in A, a < x \wedge x < b$

En d'autres mots, il est possible d'identifier dans un ensemble, même infini, une partie non ponctuelle de celui-ci transitivement close. Le graphe illustre parfaitement la relation de couverture sur les entiers (3.1). La structure ordonnée $(\mathbb{N},\leq)$ admet donc une représentation sous la forme d'un graphe où la relation de couverture est la restriction aux couples irréflexifs de la relation.

La structure ordonnée des nombres rationels $(\mathbb{Q},\leq)$ n'admet pas de représentation graphique, on peut exposer naturellement un contre-exemple. En effet, posons $a,b,c,d \in \mathbb{N}$ tels que :

$$\frac{a}{b} < \frac{c}{d} \quad \frac{a}{b}, \frac{c}{d} \in \mathbb{Q} \tag{3.54}$$

possède toujours un élément dans sa fermeture transitive :

$$\frac{a}{b} < \frac{1}{2} \times \frac{ad+bc}{bd} < \frac{c}{d} \tag{3.55}$$

En partant de $\frac{1}{2}, 1 \in \mathbb{Q}$, on obtient :

$$\frac{1}{2} < 1 \tag{3.56}$$
$$\frac{1}{2} < \frac{3}{4} < 1 \tag{3.57}$$
$$\frac{1}{2} < \frac{5}{8} < \frac{3}{4} < \frac{7}{8} < 1 \tag{3.58}$$
$$\frac{1}{2} < \frac{9}{16} < \frac{5}{8} < \frac{11}{16} < \frac{3}{4} < \frac{13}{16} < \frac{7}{8} < \frac{15}{16} < 1 \tag{3.59}$$

La figure illustre la position relative de chaque nouveau nombre rationnel construit en moyennant chaque paire de nombres déja construits.

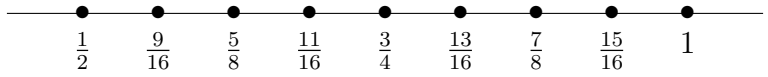


FIGURE 3.7 : Positions des rationnels sur l'intervalle réel $[\frac{1}{2}, 1]$

Définition 3.1.14 (Antitransitivité)**.** *Soit R une relation binaire sur un ensemble A, R est antitransitive si et seulement si, pour tout $a, b, c \in A$:*

$$(a, b) \in R \wedge (b, c) \in R \implies (a, c) \notin R \tag{3.60}$$

cette définition amène naturellement à exhiber les bonnes propriétés d'une relation de couverture.

Théorème 3.1.5. *Soit un ensemble ordonné $(A, \leq)$, et $<$ sa relation de couverture, alors $<$ est bien fondée et antitransitive.*

Démonstration. Par définition, $<\subseteq A \times A$ ne peut être antitransitive, puisque :

$$a < x \wedge x < b \implies x = a \vee x = b \tag{3.61}$$

De même, si $<$ n'est pas bien fondée, alors il existe une séquence $x_0, x_1, \ldots, x_n$ dans A telle que :

$$x_0 < x_1 \wedge \ldots \wedge x_{n-1} < x_n \implies x_0 = x_n \tag{3.62}$$

Or puisque $<$ est antitransitive, on a :

$$x_0 = x_1 = \ldots = x_{n-1} = x_n \tag{3.63}$$

$$\implies x_0 \not< x_n \tag{3.64}$$

ce qui aboutit à une contradiction. Donc $<$ est naturellement bien fondée. □

On peut dorénavant proposer une définition explicite du diagramme de Hasse.

Définition 3.1.15. *Soit un ensemble ordonnée $(A, \leq)$, on définit le diagramme de Hasse de A par $H(A)$ et tel que l'on dessine un trait vertical depuis x vers y si et seulement si, y couvre x.*

Les figures (3.4) et (3.6) sont des exemples valides de diagramme de Hasse.

Comme cela a été précisé tout au long de la section, les ordres expriment deux sortes de dualités : la première, mathématiquement parlant, basée sur l'inverse de l'ordre ne discrimine pas un ordre de son ordre inverse, la seconde dualité exprime qu'un ordre partiel et son ordre strict ne diffèrent pas en regard des propriétés qu'elles vérifient. Le diagramme de Hasse est donc une représentation équitable dans ces circonstances puisqu'il ne favorise pas une interprétation plutôt qu'une autre.

3.1.4 Bornes et sous-ensembles remarquables

On va poser quelques définitions indispensables à la mise en œuvre d'opérations sur des structures – partiellement – ordonnées. Toutes les définitions exposées ci-dessous sont valables lorsque l'ordre associé est total.

Définition 3.1.16 (Majorant)**.** *Soit un ensemble partiellement ordonné* $(A, \leq)$*, et une partie* B *de* A*,* $y \in A$ *est un* majorant *de* B *si et seulement si la condition suivante est vérifiée :*

$$\forall x \in B, x \leq y \tag{3.65}$$

La dualité sur les ordres induit toujours l'existence d'une définition duale.

Définition 3.1.17 (Minorant)**.** *Soit un ensemble partiellement ordonné* $(A, \leq)$*, et une partie* B *de* A*,* $m \in A$ *est un* minorant *de* B *si et seulement si la condition suivante est vérifiée :*

$$\forall x \in B, y \leq x \tag{3.66}$$

Un élément dans un ordre est alors un majorant si il couvre les éléments d'une partie de celui-ci. Inversement, un minorant est un élément qui est couvert par tous les éléments d'une partie d'un ordre.

On note que de tels éléments peuvent ne pas exister ou ne pas être uniques à l'instar de sous-ensembles d'algèbres qui ne sont pas nécessairement fermés par l'application de leur opération. Une partie est dite *majorée* si elle contient l'ensemble de ses majorants et *minorée* si elle contient l'ensemble de ses minorants.

La figure (3.8) est un exemple d'ensemble ordonné pour lequel on peut extraire des sous-ensembles majorés et des sous-ensembles minorés. Les figures (3.9) et (3.10) en présentent quelques-uns à titre d'exemple.

Définition 3.1.18 (Borne supérieure)**.** *Soit un ensemble ordonné* $(A, \leq)$ *et une partie* B *de* A*, On note* $s = \sup B$*, la borne supérieure de* B*, lorsqu'elle existe, telle que :*

1. $\forall x \in B, s \geq x$

s *est aussi appelé un* supremum *de* B*.*

Définition 3.1.19 (Borne inférieure)**.** *Soit un ensemble ordonnée* $(A, \leq)$ *et une partie* B *de* A*, On note* $i = \inf B$*, la borne inférieure de* B*, lorsqu'elle existe, de* B *telle que :*

1. $\forall x \in B, i \leq x$

i *est aussi appelé* infimum *de* B*.*

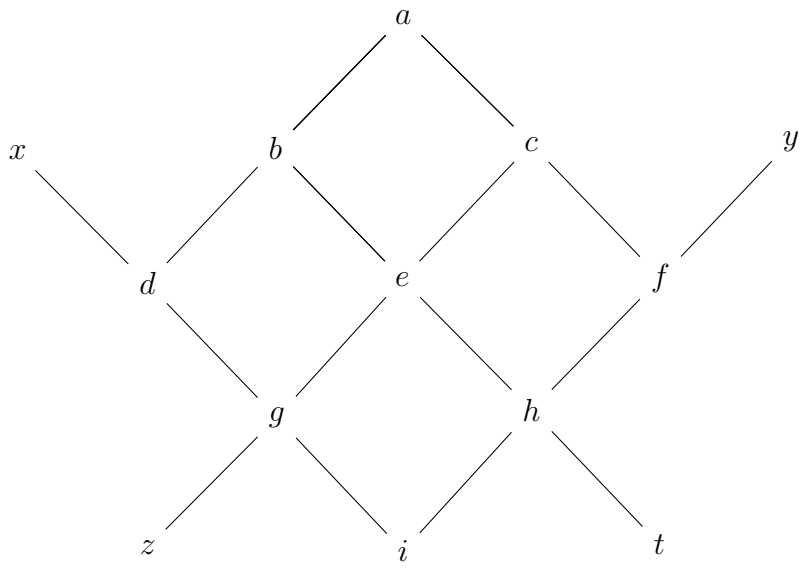


FIGURE 3.8 : Un ensemble partiellement ordonné quelconque $(A, \leq)$

Par antisymétrie de la relation d'ordre partiel, les bornes[3] lorsqu'elles existent sont nécessairement uniques. Les parties exposées précédemment sont en l'occurrence toutes munies d'une borne.

Ces définitions se généralisent aisément aux parties elles-mêmes.

Définition 3.1.20 (Partie bornée)**.** *Soit un ensemble ordonné $(A, \leq)$ et une partie B de A, B est dite bornée lorsqu'il existe simultanément dans B, un supremum et un infimum.*

La partie exposée dans la figure (3.9(d)) est bornée[4] car elle possède l'élément a comme supremum et e comme infimum.

Lorsque c'est toute la structure qui est bornée, il est d'usage d'identifier ces bornes à l'aide des symboles $\top$, désigné « taquet vers le haut », et $\bot$ pour « taquet vers le bas ».

On va s'intéresser maintenant aux parties préservant certaines propriétés relationnelles de la structure.

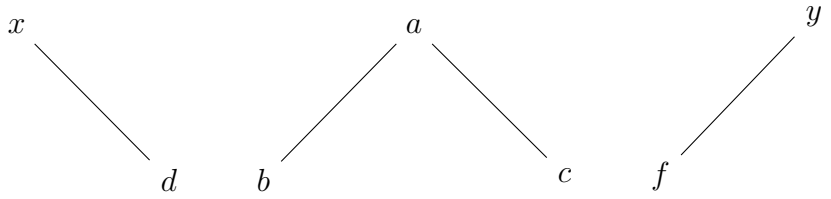
Définition 3.1.21 (Parties engendrées par l'ordre)**.** *Soit un ensemble ordonné $(A, \leq)$. On note $\uparrow x$ et $\downarrow x$, les ensembles des majorants et minorants de x dans A par :*

$$\uparrow x \triangleq \{y \in A \mid x \leq y\} \tag{3.67}$$

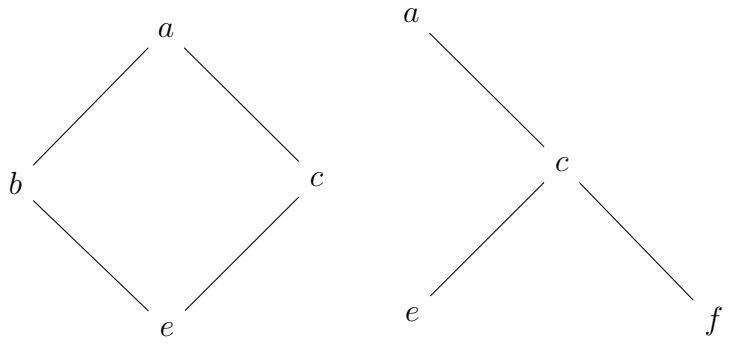
$$\downarrow x \triangleq \{y \in A \mid y \leq x\} \tag{3.68}$$

[3] Les bornes décrites ici font donc respectivement références dans la littérature anglophone aux *least upper bound* et *greatest lower bound*.

[4] Lorsqu'une partie ou la structure elle-même n'est que partiellement bornée, il n'existe pas d'épithète adéquat permettant de qualifier une telle observation à l'instar de ceux utilisés par les anglophones comme *upper bounded* et *lower bounded*. Une partie bornée servira uniquement par la suite pour décrire le cas « complet ».

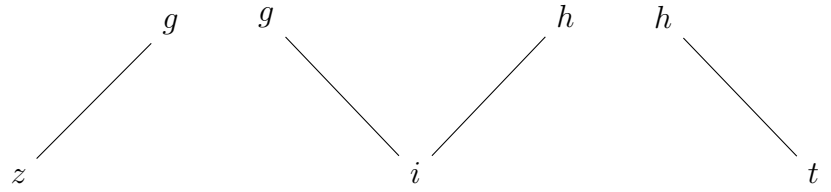


(a) $X = \{x, d\}$, $\sup X = \{x\}$ (b) $X = \{a, b, c\}$, $\sup X = \{a\}$ (c) $X = \{f, y\}$, $\sup X = \{y\}$



(d) $X = \{a, b, c, e\}$, $\sup X = \{a\}$ (e) $X = \{a, b, d, x\}$, $\sup X = \{a\}$

FIGURE 3.9 : Quelques exemples de parties majorées X de la structure (3.8)



(a) $X = \{g, z\}$, $\inf X = \{z\}$ (b) $X = \{g, h, i\}$, $\inf X = \{i\}$ (c) $X = \{h, t\}$, $\inf X = \{t\}$

FIGURE 3.10 : Quelques exemples de parties minorées X de la structure (3.8)

$\uparrow x$ *est souvent désigné par* filtre principal engendré par x *tandis que* $\downarrow x$ *désigne l'*idéal principal engendré par x[5].

Ces définitions s'étendent naturellement à toute partie X non-vide d'un treillis L :

$$\uparrow X \triangleq \{y \in A \mid \exists x \in X, x \leq y\} \tag{3.69}$$

$$\downarrow X \triangleq \{y \in A \mid \exists x \in X, y \leq x\} \tag{3.70}$$

En revanche, contrairement aux filtres et idéaux principaux, ce ne sont pas des parties

[5]Le lecteur aura remarqué que ces dénominations font référence à des sous-ensembles remarquables, que l'on rencontre le plus souvent en théorie des groupes et des anneaux. En effet, on verra dans la description algébrique de l'ordre par le biais des treillis la raison de cet emprunt et donc des connexions entre ces théories.

fermées, respectivement, pour la borne inférieure et la borne supérieure. On admet généralement la dénomination de cône positif et de cône négatif, dans ce cas [47, p. 119-120].

Ces parties permettent de définir la notion d'intervalle entre deux éléments $[x, y]$ *lorsque* $x \leq y$ *avec :*

$$[x, y] \triangleq \{z \in A \mid x \leq z, z \leq y\} \tag{3.71}$$
$$= \uparrow x \cap \downarrow y \tag{3.72}$$

qu'on retrouve également sous la forme de y/x *dans la littérature.*

Définition 3.1.22 (Sous-ensemble partiellement ordonné)**.** *Soit un ensemble ordonnée* $(A, \leq)$*, et une partie* B *de* A*. On appelle* B *un sous-ensemble partiellement ordonné de* A *(abrégé* sous-epo[6]*) si pour tout* x, y *dans* B *:*

$$x \leq_B y \implies x \leq y \tag{3.73}$$

En l'absence d'ambiguité dans la notation, on omettra le plus souvent la restriction à B *en indice de la relation d'ordre.*

Par exemple, pour tout élément x, filtres et idéaux sont donc des parties fermées pour l'ordre et possèdent une grande importance dans la théorie de la représentation des treillis.

Lemme 3.1.3. *Soit un epo* $(A, \leq_A)$ *muni d'un infimum et un sous-epo* $(B, \leq)$*; si* B *possède un infimum alors :*

$$\inf(A) \leq_A \inf(B) \tag{3.74}$$

Démonstration. Par définition de l'inclusion, on a :

$$x \in B \implies x \in A \tag{3.75}$$
$$\wedge\, x \in A \implies \inf(A) \leq_A x \qquad \text{(def.)} \tag{3.76}$$
$$\Leftrightarrow x \in B \implies \inf(A) \leq_A x \tag{3.77}$$

Donc $\inf(A) \leq \inf(B)$ car $\inf(B) \in B$, ce qui conclut la preuve. □

On obtient un résultat similaire dans le cas du supremum.

Corollaire 3.1.1. *Soit un epo* $(A, \leq)$ *et un sous-epo* $(B, \leq)$ *de* A*. Soit une partie quelconque* S *de* A *telle que* $i \in S$ *et* $i = \inf(B)$*, alors* $i = \inf(B \cap S)$*.*

On voit facilement que l'intersection de plusieurs ordres ne remet pas en cause leur borne supérieure ou inférieure.

[6] Toute référence à Richard Virenque est purement fortuite...

Définition 3.1.23 (Section commençante et finissante)**.** *Soit un epo* $(A, \leq)$ *et une partie* B *de* A*,* B *est une* section commençante *de* A *si et seulement si :*

$$\forall a \in B, \forall x \in A, x \leq a \implies x \in B \tag{3.78}$$

Dualement, B *est une* section finissante *de* A *lorsque :*

$$\forall a \in B, \forall x \in A, a \leq x \implies x \in B \tag{3.79}$$

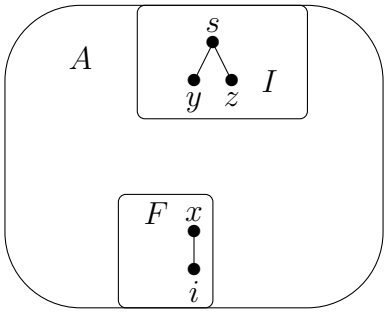


FIGURE 3.11 : Illustration d'un idéal $I \subset A$ et d'un filtre $F \subset A$

Lemme 3.1.4. *Le filtre* $\uparrow x$ *est une section finissante tandis que l'idéal* $\downarrow x$ *est une section commençante.*

La preuve est omise, il suffit d'observer que ces parties sont fermées pour l'ordre et que la définition de ces sections est précisément effectuée sur des conditions de fermeture pour l'ordre : une section commençante est une partie minorée pour l'ordre et dualement pour une section finissante.

En raison de cette connexion, ces sections sont souvent simplement désignées par *filtre* et *idéal* de l'ordre lorsqu'elles sont bornées. La figure (3.11) illustre la notion d'idéal et de filtre d'un ensemble ordonné.

Définition 3.1.24 (Filtre-idéal maximal)**.** *Soit un epo* $(A, \leq)$ *et un filtre* $F \subset A$*,* F *est dit maximal si et seulement si il existe un filtre* $G \subset A$ *telle que :*

$$F \subseteq G \implies G = F$$

Un idéal maximal est obtenu de manière analogue.

On notera par la suite $\mathcal{F}(A)$ et $\mathcal{I}(A)$, les ensembles des filtres et idéaux d'un ensemble ordonnée $(A, \leq)$. La figure (3.12) illustre l'ordre induit par l'inclusion pour la collection des sous-groupes d'un groupe tandis que la table (3.1.4) renvoie aux ensembles des filtres et idéaux maximaux pour l'ordre associé, soit $\mathcal{F}_{max}$ et $\mathcal{I}_{max}$ hormis les triviaux $\uparrow 0$ et $\downarrow 1$.

On remarque aisément que les filtres et idéaux principaux sont des parties génératrices remarquables d'un treillis A. Nous verrons par la suite que celles-ci permettent de

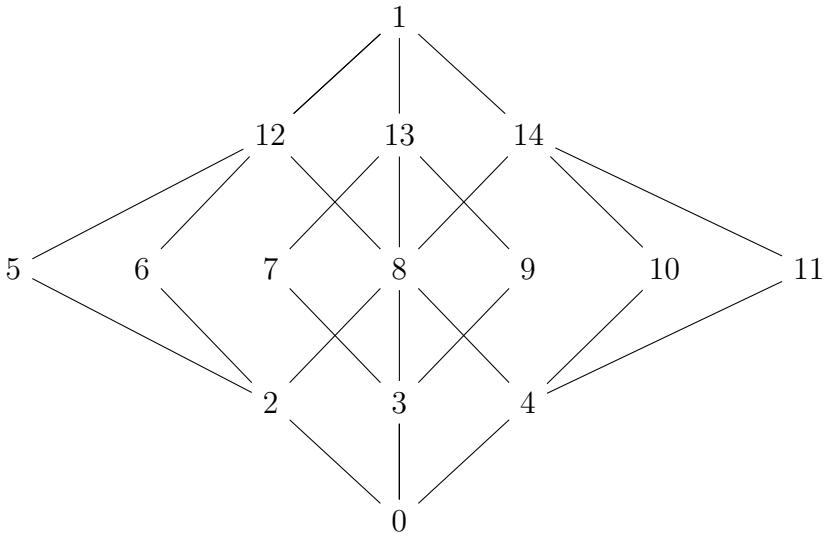


FIGURE 3.12 : Treillis engendré par l'inclusion des sous-groupes.

$\mathcal{F}_{max}$	$\mathcal{I}_{max}$
$\uparrow 2$	$\downarrow 12$
$\uparrow 3$	$\downarrow 13$
$\uparrow 4$	$\downarrow 14$

TABLE 3.2 : Les filtres et idéaux maximaux de l'ordre (3.12)

décrire chaque élément $x \in A$ sous la forme d'un ensemble d'éléments $X \in \mathcal{P}(A)$ de la même structure. Une construction similaire est proposé par l'équation (3.17) qui établit la correspondance entre certains entiers et un ensemble de facteurs premiers (*c.f.* figure 3.4).

Nous allons maintenant introduire une classe spéciale d'ensembles partiellement ordonnés pour laquelle, infimum et supremum existent pour tout couple d'éléments.

3.2 Treillis et algèbre de l'ordre

Avant d'introduire formellement les treillis, on va discuter de la construction de l'ensemble des parties d'un ensemble X, soit $\mathcal{P}(X)$.

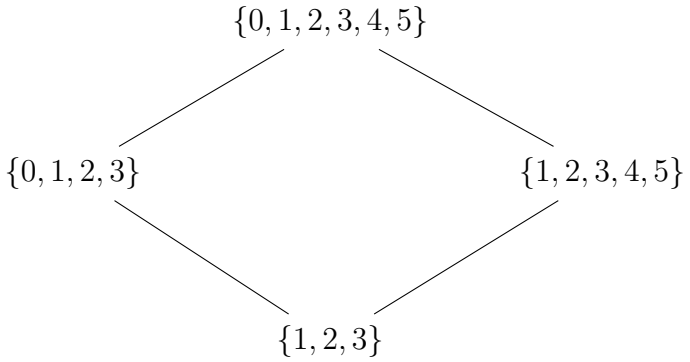


FIGURE 3.13 : Infimum et Supremum des ensembles $\{0, 1, 2, 3\}$ et $\{1, 2, 3, 4, 5\}$

3.2.1 Préliminaires

Lemme 3.2.1. *Soit une collection d'ensembles* $\{X_i\} \in \mathcal{P}(X)$*, alors on a :*

$$\inf \{X_i\} = \bigcap_i X_i \tag{3.80}$$

$$\sup \{X_i\} = \bigcup_i X_i \tag{3.81}$$

Démonstration. Par la dualité de l'ordre, prouver l'existence de l'infimum est suffisant. En reprenant la même méthodologie que pour les familles d'algèbres (*c.f.* 2.2.1), il suffit de démontrer :

$$\forall X_i, S \subseteq X_i \Leftrightarrow S \subseteq \bigcap_i X_i \tag{3.82}$$

On pose $S \subseteq X_1$ et $S \subseteq X_2$, alors :

$$x \in S \implies x \in X_1 \wedge x \in X_2 \tag{3.83}$$

$$\implies x \in X_1 \cap X_2 \tag{3.84}$$

$$\implies S \subseteq X_1 \cap X_2 \tag{3.85}$$

Donc $S \subseteq X_1$ et $S \subseteq X_2 \implies S \subseteq X_1 \cap X_2$.

Inversement, posons $S \subseteq X_1 \cap X_2$, puisque la relation est transitive, on a directement $S \subseteq X_1$ et $S \subseteq X_2$. Donc, on a $S \subseteq X_1$ et $S \subseteq X_2 \Leftrightarrow S \subseteq X_1 \cap X_2$. par associativité de l'intersection, on a alors :

$$S \subseteq \bigcap X_i \Leftrightarrow S \subseteq X_i, \forall_i \tag{3.86}$$

$\bigcap X_i$ est donc nécessairement la borne inférieure commune à chaque X_i, ce qui complète la preuve. □

Corollaire 3.2.1. *Soit une collection d'ensembles $\mathcal{X} = \{X_i\}$, l'ensemble de toutes les intersections et unions forme un treillis complet avec l'inclusion comme relation d'ordre, la borne inférieure étant la plus grande intersection et la borne supérieure, la plus grande union.*

En utilisant de nouveau le corollaire (3.1.1), soit la sous-famille $\mathcal{Y} \subset \mathcal{X}$, alors $\bigcap \mathcal{X} \subset \bigcap \mathcal{Y}$, et réciproquement pour l'union. Les bornes inférieure et supérieure de chaque opération forme alors un ordre partiel et par réciprocité, l'ensemble des opérations sur la structure a pour résultat une borne de celle-ci.

On peut vérifier sur la figure (3.14) que le treillis engendré sur une partie de $\mathcal{P}(X)$ avec $X = \{a, b, c, d\}$ est complet. Les diagrammes (3.14(b)-3.14(g)) présentent l'ensemble des décompositions binaires propres à l'obtention de toutes les bornes du treillis. En particulier, cela montre que l'opération de construction des bornes est indépendante de l'arité choisie.

Dans un contexte algébrique, l'infimum de deux éléments dans un treillis sera noté en écriture symbolique $a \wedge b$ et le supremum sera lui noté $a \vee b$. Le choix du symbole n'est lui-même pas sans rappeler ceux de la conjonction et de la disjonction logique. Pour des raisons historiques, les modèles d'interprétations algébriques des logiques sont traditionnellement des treillis pour lesquelles le terme anglophone de *meet* pour la conjonction signifie que les propositions ou phrases afférentes aux opérandes impliquent que le modèle associé à la proposition résultante « suit » ou « rencontre » celles de ses prémisses. À l'opposé, le modèle associé à la disjonction construit la représentation du modèle en « unifiant » ou « joignant » les conditions propres à chacune de ses prémisses.

Jusqu'ici on a insisté sur les propriétés ordinales propres à la structure de treillis liées à l'existence de l'une ou l'autre borne. On va maintenant caractériser les différentes classes de treillis et la nature des opérations que l'on peut construire sur elles.

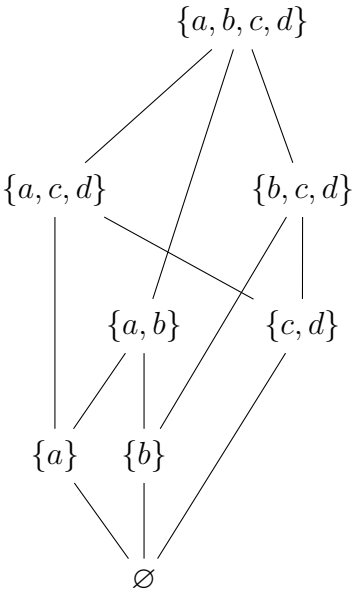
3.2.2 Propriétés ordinales des treillis

Définition 3.2.1 (Inf/sup-demi-treillis)**.** *Soit un epo $(A, \leq)$. Alors A est un* inf-demi-treillis *si et seulement si toute paire d'éléments $a, b \in A$ possède une borne inférieure telle que $a \wedge b = \inf\{a, b\}$. De même, A est un* sup-demi-treillis *si et seulement si toute paire d'éléments possède une borne supérieure telle que $a \vee b = \sup\{a, b\}$.*

Par la dualité de l'ordre associé à la structure, il est souvent d'usage d'abréger la caractérisation en demi-treillis, lorsque la différentiation n'a pas d'importance.

Définition 3.2.2 (Treillis)**.** *Soit un epo $(A, \leq)$. Alors A est un treillis, si A est simultanément un inf-demi-treillis et un sup-demi-treillis.*

La figure (3.14) est le diagramme de Hasse d'un treillis puisque chaque paire d'éléments possède une borne supérieure et une borne inférieure. Cependant, tous les ordres



(a) Treillis engendré par la collection $\{\{a,b\},\{a,c,d\},\{b,c,d\}\} \subseteq \mathcal{P}(X)$

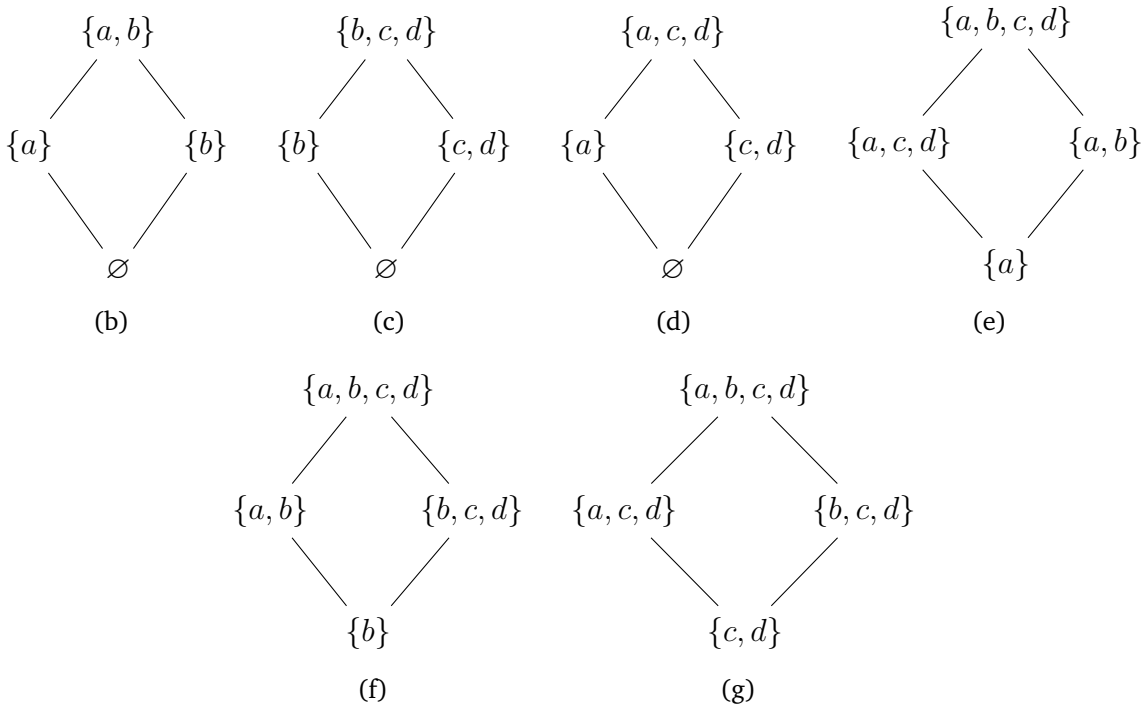


FIGURE 3.14 : Le diagramme de Hasse en haut figure l'inclusion de toutes les bornes. Chaque sous-diagramme représente une opération binaire non-triviale.

partiels fermés pour l'ordre, c.-à-d. munis de supremum et d'infinum, ne sont pas nécessairement des treillis.

La figure (3.15) présente la décomposition de l'entier 5 sous la forme d'une séquence

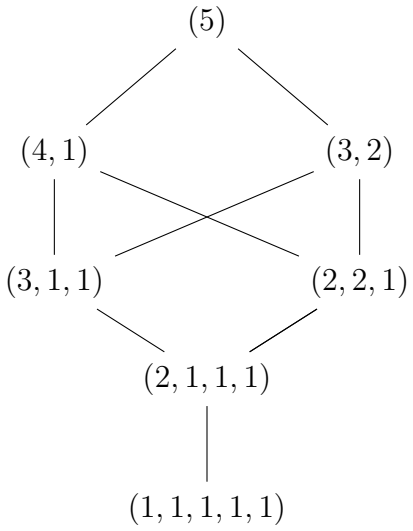


FIGURE 3.15 : Le diagramme de Hasse des décompositions de l'entiers 5.

ordonnée d'entiers à sommer. En notant deux décompositions $a \triangleq (a_1, a_2, \ldots)$ et $b \triangleq (b_1, b_2, \ldots)$ telles que $\forall i, a_i \geq a_{i+1}$ et $b_i \geq b_{i+1}$, on a l'identité suivante :

$$x \lesssim y \Leftrightarrow \forall i > 1, \sum_i a_i \leq \sum_i b_i \tag{3.87}$$

En particulier, on constate que $(3,1,1)$ et $(2,2,1)$ possèdent deux bornes supérieures donc $(3,1,1) \vee (2,2,1)$ n'est pas une opération algébrique et ce n'est donc pas un demi-treillis. Par réciprocité, le même constat vaut pour $(4,1)$ et $(3,2)$ qui ont pour bornes inférieures $(3,1,1)$ et $(2,2,1)$.

On va maintenant caractériser l'équivalence entre les propriétés de l'ordre partiel et les identités algébriques d'un treillis.

Théorème 3.2.1. *Soit un treillis $(A, \wedge, \vee)$, on a l'identité suivante pour tout $a, b \in A$:*

$$a \leq b \Leftrightarrow \begin{cases} a \vee b = b \\ a = a \wedge b \end{cases} \tag{3.88}$$

Démonstration. On pose $a, b \in A$, on a alors :

$$a \leq b \implies a \leq b \wedge b \leq b \qquad \text{(refl.)} \tag{3.89}$$

$$\tag{3.90}$$

et b est un majorant de a et de lui-même. Soit $c \in A$ tel que c est un majorant de a et b : $a \leq c \wedge b \leq c$. Par $b \leq c$, b est nécessairement le plus petit majorant de $\{a, b\}$ donc $a \vee b = b$. L'argument de l'ordre permet de démontrer la propriété analogue pour $a = a \wedge b$.

On présume maintenant que pour tout $a, b \in A$, $a \vee b$ existe dans le treillis. Dans ce cas, $a \vee b = b$ implique $b = a \vee b \geq a$ par définition de la borne supérieure d'un ordre, ce qui complète la preuve. □

Définition 3.2.3 (Treillis complet[7])**.** *Soit un epo $(A, \leq)$. Alors A est un treillis, si toute partie non-vide (possiblement infinie) S de A possède une borne inférieure et supérieure.*

En gardant à l'esprit que le même treillis est *par construction* le résultat de toutes les unions et intersections d'ensembles possibles depuis ceux fournis au préalable, la structure associée est donc par définition complète bien que ce soit une partie stricte de $\mathcal{P}(X)$. En particulier, on pourra démontrer qu'il existe un univers Y dans lequel cette structure est isomorphe à $\mathcal{P}(Y)$.

Théorème 3.2.2. *Un treillis complet A est nécessairement borné.*

Démonstration. Les bornes correspondantes aux taquets $\top$ et $\bot$ sont respectivement obtenues par le calcul de la borne supérieure et de la borne inférieure sur tout A. □

$\mathcal{P}(X)$ est un treillis complet et donc borné. Les rationnels $\mathbb{Q}$ bien que ne possédant pas de représentation graphique formelle ne possèdent ni borne inférieure ni supérieure à l'instar de $\mathbb{R}$, même en considérant des parties infinies.

Plus simplement, les entiers naturels $\mathbb{N}$ possèdent une représentation finie sur un intervalle où les bornes sont définies par le calcul du maximum et du minimum mais ne possède pas de bornes minimale et maximale. $\mathbb{N} \cup \{+\infty\}$ est en revanche un treillis complet et borné.

L'ordre sur la divisibilité, schématisé par la figure (3.6), est un treillis complet où $a \vee b$ est défini par le *plus petit multiplicateur commum* (ppcm) et $a \wedge b$ est défini par le *plus grand diviseur commun* (pgcd).

De même, le treillis des partitions est un treillis complet $(\Pi^{\Omega}, \leq, \wedge, \vee)$. Celle-ci est munie d'une relation d'ordre sur les partitions.

Définition 3.2.4 (Raffinement)**.** *Soit le treillis des partitions $(\Pi_{\Omega}, \leq, \wedge, \vee)$, on dit que P raffine Q si et seulement si, on a :*

$$\forall C \in P,\ \exists C' \in Q, C \subseteq C' \tag{3.91}$$

notée, $P \leq Q$.

Les opérateurs latticiels admettent les définitions concrètes suivantes.

Définition 3.2.5. *Soit le treillis des partitions $(\Pi_{\Omega}, \leq, \wedge, \vee)$, étant données deux partitions P, Q, on a :*

$$P \wedge Q = \{C \cap C' \mid C \in P, C' \in Q, C \cap C' \neq \varnothing\} \tag{3.92}$$
$$P \vee Q = \{\inf D \mid C \cup C' \subseteq D, C \in P, C' \in Q, C \cap C' \neq \varnothing\} \tag{3.93}$$

notée, $P \leq Q$.

[7] Au sens de Birkhoff. Bourbaki proposa l'épithète « achevé » qui ne fut pas retenue bien que plus explicite.

Théorème 3.2.3. *Soit un epo $(A, \leq)$, Si toute partie S (même vide) de A possède un supremum dans A, alors A est un treillis complet.*

Démonstration. Dans la mesure où un treillis est complet si toutes les bornes sont définies, il suffit de prouver l'existence de la borne inférieure pour toute partie X de A. Sans perte de généralité, on part du principe que $X = \{a, b\}$. Puisque A est, par définition, inférieurement borné, $\forall a, b \in A, \bot \leq a, \bot \leq b$. Par définition de la borne supérieure, a et b sont nécessairement des majorants de l'ordre tel que $\{x \in A \mid x \leq a \wedge x \leq b\}$. En particulier, a et b majorent nécessairement la *plus petite* borne supérieure tels que :

$$\bigvee \{x \in A \mid x \leq a, x \leq b\} \leq \begin{cases} a \\ b \end{cases} \tag{3.94}$$

$$\Leftrightarrow a \wedge b \leq \begin{cases} a \\ b \end{cases} \tag{3.95}$$

On en déduit que la borne inférieure de toute partie X existe dans le treillis, qui est par conséquent complet, ce qui complète la preuve. □

Une conséquence immédiate du précédent théorème est que chaque opérateur du treillis lorsqu'il est complètement défini, permet de déduire le second.

Corollaire 3.2.2. *Soit un treillis complet $(A, \leq, \vee, \bot, \top)$, on définit la borne inférieure d'une partie X de A par :*

$$\bigwedge X \triangleq \bigvee \{y \mid \forall x \in X, y \leq x\} \tag{3.96}$$

En combinant ces deux résultats, on obtient l'équivalence suivante entre définitions d'un treillis complet :

Théorème 3.2.4. *Soit un epo $(A, \leq)$. Les conditions suivantes sont équivalentes :*

1. *$(A, \leq, \wedge, \vee, \bot, \top)$ est un treillis complet*
2. *A possède une borne supérieure $\top$ et $\bigwedge X$ existe pour toute partie X de A.*

Démonstration. Il est facile de montrer qu'un treillis complet implique l'existence des bornes inférieures. En effet, $\top = \bigvee A = \bigwedge \varnothing$ et $\bigwedge X$ sont dans A pour toute partie X de A.

À l'inverse, l'existence des bornes inférieures $\bigwedge X$ implique nécessairement l'existence des bornes supérieures dans $\bigvee X$ en utilisant le même procédé que dans le précédent théorème. On pose $X = \{a, b\}$, on a alors :

$$S = \{y \in A \mid a \leq y, b \leq y\} \tag{3.97}$$

Par définition d'un majorant, $\forall y \in S, y \geq a$ et $y \geq b$ donc $\bigvee S$ est également un majorant. Soit s un majorant, on a $a \leq s$ et $b \leq s$ donc $s \in S$. Ceci implique $\bigvee S \leq s$, donc $\bigvee S$ est la borne supérieure de a et b.

Puisque $\top \in A$, $\bigvee A = \top$, on a donc :

$$a \vee b = \bigwedge \{y \in A \mid a \leq y, b \leq y\} \tag{3.98}$$

et donc l'existence de toutes les bornes inférieures implique l'existence des bornes supérieures. □

On va illustrer cette méthodologie en détaillant la construction du treillis de la figure (3.14(a)). Posons $X = \{a, b\}, Y = \{a, c, d\}, Z = \{b, c, d\}$. On procède d'abord en engendrant l'ensemble des bornes inférieures associées et correspondant à l'opération ensembliste d'intersection :

$$\begin{aligned}
\bigcap \{X, Y, Z\} &= \varnothing \\
\bigcap \{X, Y\} &= \{a\} \\
\bigcap \{X, Z\} &= \{b\} \\
\bigcap \{Y, Z\} &= \{c, d\} \\
\bigcap \{X, X\} &= X \\
\bigcap \{Y, Y\} &= Y \\
\bigcap \{Z, Z\} &= Z \\
\bigcap \varnothing &= \{a, b, c, d\}
\end{aligned} \tag{3.99}$$

Suivant la définition équivalente de la borne supérieure, l'union de deux ensembles s'obtient alors par l'intersection de tous leurs majorants :

$$\begin{aligned}
\{a\} \cup \{b\} &= \bigcap \{x \in A \mid a \leq x, b \leq x\} \\
\Leftrightarrow \{a\} \cup \{b\} &= (X \cup Y) \cap (X \cup Z) = X = \{a, b\}
\end{aligned} \tag{3.100}$$

On retrouve dès lors facilement les autres bornes supérieures :

$$\begin{aligned}
\{a\} \cup \{c, d\} &= (X \cup Y) \cap (Y \cup Z) = Y = \{a, c, d\} \\
\{b\} \cup \{c, d\} &= (X \cup Z) \cap (Y \cup Z) = Z = \{b, c, d\} \\
\{a, b\} \cup \{c, d\} &= X \cap (Y \cup Z) = \varnothing = \{a, b, c, d\}
\end{aligned} \tag{3.101}$$

Il est possible d'interpréter cette équivalence comme conséquence au fait que l'ordre partiel $(A, \leq)$ est par définition bien fondé. En effet, celle-ci équivaut à dire que toute partie non-vide contient un élément minimal pour l'ordre – ou dualement maximal. Toute séquence d'éléments $x_1 \geq x_2 \geq \ldots$ est alors nécessairement stationnaire pour un entier $n \in \mathbb{N}$ tel que $x_n = x_{n+1}$ et figure donc une borne inférieure[8].

Théorème 3.2.5. *Soit un treillis complet $(A, \leq, \vee, \wedge)$, les conditions suivantes sont équivalentes :*

[8] condition de chaîne descendante pour *descending chain condition* dans la littérature anglophone

1. *A a une borne supérieure $\top$ et vérifie la condition de chaîne descendante*
2. *A n'a pas de séquence d'éléments infinie pour l'ordre*

On va maintenant s'intéresser aux propriétés algébriques des treillis dont on va donner une définition générale.

Définition 3.2.6 (Treillis)**.** *Soit un ensemble A, la structure $(A, \wedge, \vee, \bot, \top)$ est un treillis si et seulement si les lois suivantes sont vérifiées pour tout $a, b, c \in A$.*

Lois de commutativité :

$$a \wedge b = b \wedge a \quad a \vee b = b \vee a \tag{3.102}$$

Loi d'associativité :

$$a \wedge (b \wedge c) = (a \wedge b) \wedge c \quad a \vee (b \vee c) = (a \vee b) \vee c \tag{3.103}$$

Lois d'absorption :

$$a \vee (a \wedge b) = a \quad a \wedge (a \vee b) = a \tag{3.104}$$

Lois idempotentes :

$$a \vee a = a \quad a \wedge a = a \tag{3.105}$$

Les identités :

$$a \vee \bot = a \quad a \wedge \top = a \tag{3.106}$$

On distingue généralement les deux classes de treillis suivantes et vérifiant des axiomes supplémentaires.

Définition 3.2.7 (Distributivité)**.** *Soit un treillis $(A, \wedge, \vee)$, A est un treillis distributif si et seulement si il vérifie pour tout $a, b, c \in A$ les conditions suivantes :*

$$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c) \tag{3.107}$$
$$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c) \tag{3.108}$$

Définition 3.2.8 (Modularité)**.** *Soit un treillis $(A, \wedge, \vee)$, A est un treillis modulaire si et seulement si il vérifie pour tout $a, b, c \in A$ les conditions suivantes :*

$$a \vee (b \wedge c) = (a \vee b) \wedge c \tag{3.109}$$
$$a \wedge (b \vee c) = (a \wedge b) \vee c \tag{3.110}$$

Il est intéressant de voir que la propriété de distributivité nécessite également de vérifier celle de modularité (sans être réciproque toutefois).

Pour terminer cette section, nous ajoutons une propriété caractéristique des treillis dont l'ordre sous-jacent est bien fondé – toutes les chaînes ont alors la même taille.

Définition 3.2.9 (Fonction de rang)**.** *Soit un treillis $(A, \leq, \wedge, \vee)$ muni d'une fonction de* rang $\mathrm{rk} : A \to \mathbb{N}$ *et vérifie les conditions suivantes pour tout $a, b \in A$:*

$$a < b \Rightarrow \mathrm{rk}(a) < \mathrm{rk}(b) \tag{3.111}$$
$$\nexists x \in A, a < x \wedge x < b \Rightarrow \mathrm{rk}(b) = \mathrm{rk}(a) + 1 \tag{3.112}$$

3.2.3 Morphisme de treillis

On va s'intéresser aux propriétés des fonctions opérant sur des ordres en tant que morphisme préservant des propriétés liées à l'ordre – dans une veine comparable au chapitre précédent – et leur généralisation dans le contexte algébrique des treillis.

Définition 3.2.10 (Homomorphisme de treillis)**.** *Soient* $(A, \wedge_A, \vee_A)$ *et* $(B, \wedge_B, \vee_B)$*, deux treillis, et une fonction* $h : A \to B$*.* h *est un* homomorphisme de treillis *si et seulement si pour tout* $a, b \in A$ *:*

$$h(a \wedge_A b) = h(a) \wedge_B h(b) \tag{3.113}$$
$$h(a \vee_A b) = h(a) \vee_B h(b) \tag{3.114}$$

h *est de plus* complet *si il préserve les bornes telles que :*

$$h(\bot_A) = \bot_B \tag{3.115}$$
$$h(\top_A) = \top_B \tag{3.116}$$

Cette définition correspond en tout point à la définition (2.3.3) donnée dans le cadre d'une structure algébrique abstraite. On distingue néanmoins à partir de cette définition deux autres classes de morphismes de treillis.

Définition 3.2.11 (homomorphisme de inf demi-treillis)**.** *Soient deux inf demi-treillis* $(A, \wedge_A)$ *et* $(B, \wedge_B)$*, et une fonction* $h : A \to B$*.* h *est un homomorphisme de inf demi-treillis si et seulement si pour tout* $a, b \in A$ *:*

$$h(a \wedge_A b) = h(a) \wedge_B h(b) \tag{3.117}$$

h *est de plus complet si* $h(\top_A) = \top_B$*.*

La définition duale s'obtient aisément pour les sup demi-treillis en substituant la borne inférieure avec la borne supérieure. L'épithète complet conserve le même sens pour les morphismes que celle employée pour qualifier un treillis dont les parties admettent des bornes. Ce qui induit la nécessité de préserver les taquets d'une structure à l'autre sous l'application du morphisme, en sus.

Lorsqu'il sera nécessaire de distinguer des morphismes qui ne sont pas stables pour chaque opérateur, on emploiera la version francisée des termes anglophones, soient $\vee$-morphisme et $\wedge$-morphisme, pour les désigner.

Définition 3.2.12 (Monotonie)**.** *Soient* $(A, \leq)$ *et* $(B, \leq)$*, deux treillis, et une application* $f : A \to B$*. Pour tout* $a, b \in A$*, on dit que* f *est* isotone *si et seulement si :*

$$a \leq b \Rightarrow f(a) \leq f(b) \tag{3.118}$$

Inversement, f *est dite* antitone *lorsque :*

$$a \leq b \Rightarrow f(b) \leq f(a) \tag{3.119}$$

3.3 Représentation des treillis

Cette section concluera cette partie sur les treillis. Le principal intérêt lié à la construction d'une représentation analogue d'un treillis est conceptuel. Elle permet d'étudier la possibilité de manipuler des structures complexes et leurs instances à l'aide d'une représentation ensembliste donc plus facilement concevable concrètement.

En conséquence de quoi, la première interrogation concerne la possibilité de minimiser le nombre d'informations nécessaires pour décrire les éléments, dont la dimension va influencer les algorithmes nécessitant l'instanciation et l'exploration de ces structures.

Un second usage concerne l'étude des propriétés de primalité et de redondance qui caractérisent les représentations factorisées des éléments, puis les propriétés liées aux parties stables. De plus, cela influence généralement la définition et l'interprétation des opérations selon ces propriétés [41, 88, 95]. La question est d'autant plus essentielle lorsque la structure en question est le modèle d'interprétation utilisé pour raisonner (*c.f.* Section 2.4).

Les parties remarquables d'un treillis qui ont été définies dans la section précédente, joueront un rôle analogue aux sous-algèbres présentées dans le précédent chapitre, et nous permettront d'établir les interactions entre les opérateurs $(\vee)$ et $(\wedge)$ (*c.f.* également [18, 109]).

3.3.1 Éléments premiers et parties génératices

Étant donné une treillis L et un élément x, la recherche d'une définition alternative dépend principalement de la recherche d'une sous-expression $x = a \vee b$ ou $x = a \wedge b$ où x n'est ni a ni b.

Travaillant principalement sur des structures rangées, la décomposition de x induit que la représentation de chaque composante est *a priori* plus économe que celle de x seul. On a donc la définition suivante pour les éléments indécomposables.

Définition 3.3.1 ($\vee$-irréductible)**.** *Soit un treillis L et un élément x. x est $\vee$-irréductible si et seulement si la condition suivante est vérifiée :*

$$x = a \vee b \implies x = a \text{ ou } x = b \tag{3.120}$$

L'ensemble des éléments premiers pour l'opérateur $(\vee)$ forme l'ensemble $\mathcal{J}(L)$.

Les définitions duales permettent l'élaboration d'un ensemble similaire pour les éléments indécomposables pour l'opérateur $(\wedge)$, et regroupés dans l'ensemble $\mathcal{M}(L)$. De plus, les $\vee$-irréductibles j forment les filtres principaux et maximaux $\uparrow j$ du treillis, tandis que les $\wedge$-irréductibles m sont les idéaux principaux et maximaux $\downarrow m$.

Un corollaire de la définition précédente est qu'un élément est premier pour l'opérateur $(\vee)$ si et seulement si il ne couvre qu'un seul élément. Réciproquement, un élément

l'est pour l'opérateur $(\wedge)$ si et seulement si il n'est couvert que par un seul élément dans la structure.

Définition 3.3.2 (Parties fermées)**.** *Soit un treillis $(L, \leq, \wedge, \vee)$, étant donné l'élément x, on a les propriétés suivantes :*

$$a \in \uparrow x \text{ et } b \in \uparrow x \Rightarrow a \wedge b \in \uparrow x \tag{3.121}$$
$$a \in \downarrow x \text{ et } b \in \downarrow x \Rightarrow a \vee b \in \downarrow x \tag{3.122}$$

On notera qu'un filtre et un idéal n'ont pas besoin d'être principaux pour être en accord avec cette définition.

Définition 3.3.3 (Treillis (co)atomistique[9])**.** *Étant donné un treillis L, L est dit* atomistique *(*resp. coatomistique*) si et seulement si pour tout $j \in \mathcal{J}(L)$ (*resp. $m \in \mathcal{M}(L)$*), j est un* atome *tel que $\bot$ est couvert par j (*resp. m *est un* coatome *tel que m est couvert par $\top$).*

Le treillis des parties est un exemple de treillis atomistique et coatomistique, où chaque singleton $\{x\}$ est un élément $\cup$-irréductible et couvre $\varnothing$. Le treillis de la relation de divisibilité est un contre-exemple : par exemple, 4 n'est pas un nombre premier, cependant il ne peut être obtenu par la borne supérieure d'un ensemble d'atomes, soit par $4 = a \times b$ tel que $a \neq b$ donc 4 est $\vee$-irréductible sans pour autant être un atome.

Enfin, le treillis représenté par la figure (3.12) n'est ni atomistique ni coatomistique. En effet, les sous-groupes indexés par les éléments $5, 6, 7, 9, 10, 11$ sont $\vee\wedge$-irréductibles mais ne figurent ni des atomes ni des coatomes.

3.3.2 Extension canonique

Les treillis observant cette propriété permettent donc d'écrire chaque élément d'un treillis sous la forme de deux ensembles :

$$x = \bigvee_{\substack{j \in \mathcal{J}(L) \\ j \leq x}} j = \bigwedge_{\substack{m \in \mathcal{M}(L) \\ x \leq m}} m$$

En notant $\mathcal{J}_{\leq x}$ et $\mathcal{M}_{\geq x}$, les ensembles d'atomes et de coatomes de x, on a alors une définition ensembliste des opérateurs latticiels.

La construction suivante permet d'établir une correspondance entre un treillis L quelconque et son *dual* exprimé par le biais d'éléments de $\mathcal{J}(L)$ et $\mathcal{M}(L)$.

Définition 3.3.4 (Extension canonique [74])**.** *Soit un treillis L, L^δ est l'*extension canonique *de L et est définie par :*

$$h : L \to L^\delta \tag{3.123}$$
$$x \mapsto \mathcal{J}_{\leq x} \times \mathcal{M}_{\not\geq x} \tag{3.124}$$

[9] Propriété qui généralise la propriété d'atomicité d'un treillis

Cette construction est semblable à celle utilisée sur l'ensemble X pour engendrer son complété de la figure (3.14(a)) où chaque élément appartient à $\mathcal{P}(X)$ et est exprimable par une expression selon les opérateurs $(\cup, \cap)$.

Néanmoins, une caractéristique propre à cette extension est la présence de coatomes dans la représentation, à cela près que les éléments de l'ensemble $\mathcal{M}_{\not\geq x}$ sont tels que x ne les subsume pas.

La relation induite sur les atomes et coatomes ayant pour but de réaliser le principe de séparation suivant : soit un filtre F et un idéal I de telle sorte que $F \cap I = \varnothing$, alors il existe un filtre premier P tel que $F \subseteq P$ et $P \cap I = \varnothing$.

La condition de primalité ne peut cependant s'obtenir pour tous les treillis. Néanmoins, l'avantage de cette construction est de permettre la conceptualisation de l'opération de complémentation dans un treillis.

En effet, si on se place dans un contexte logique, le filtre associé à un atome représente l'ensemble des propositions démontrables tandis que l'idéal associé à un coatome représente les propositions réfutables (dont la négation peut être prouvée). Les définitions et propriétés liées à ce type d'opérateur seront présentées dans le chapitre 5 dont l'usage est notre motivation principale.

En posant la surjection d'évaluation $f : L \to \{0, 1\}$, on doit donc avoir :

$$f(x) = \begin{cases} 0 & (x \in I) \\ 1 & (x \in F) \end{cases}$$

où $F \cap I = \varnothing$ et $F \cup I = L$.

Cette fonction permet de déduire que les éléments formés par le couple filtre-idéal sont complémentaires : étant donné un élément $x \in L$, alors on a un élément atomique $j \in \mathcal{J}_{\leq x}$ et un coatome $m \in \mathcal{M}_{\not\geq x}$ tels que :

$$\begin{cases} j \wedge m = \bot \\ j \vee m = \top \end{cases}$$

Dans le cadre de cette section, nous présenterons le principe édicté par Stone et plus connu comme le *principe de séparation des filtres premiers*[10] (*c.f.* [11, 125, 127]). Cette propriété sera explicitée pour la caractérisation de la propriété de distributivité d'un treillis – on la retrouve alors généralement énoncée sous la forme d'un lemme.

Théorème 3.3.1. *Soit un treillis borné $(L, \leq, \wedge, \vee)$, pour toute paire d'éléments x, y, on a la propriété suivante :*

$$\mathcal{J}_x \cap \mathcal{J}_y = \mathcal{J}_{x \wedge y} \tag{3.125}$$
$$\mathcal{J}_x \cup \mathcal{J}_y \subseteq \mathcal{J}_{x \vee y} \tag{3.126}$$

Démonstration. Premièrement, on observe que l'inclusion est toujours satisfaite dans chaque équation en observant que h est un morphisme de treillis – préservant la relation

[10] *Boolean prime ideal/filter theorem* en version originale.

d'ordre – puis que les filtres et les idéaux sont ordonnés. L'égalité dans le cas du $(\wedge)$ est simple : étant donné $j \in \mathcal{J}_{x \wedge y}$, on a :

$$j \leq x \wedge y \tag{3.127}$$
$$\iff j \leq x \text{ et } j \leq y \tag{3.128}$$
$$\iff j \in \mathcal{J}_x \cap \mathcal{J}_y \tag{3.129}$$

□

Nous allons maintenant préciser sous quelles conditions l'égalité peut advenir dans la seconde équation, permettant de simuler l'opérateur de $(\vee)$ par le biais d'union d'atomes en utilisant le lemme de séparation.

Lemme 3.3.1 (Principe de séparation d'un filtre premier)**.** *Soit un treillis distributif L, deux éléments a, b tel que $a \not\leq b$. Il existe alors un filtre premier P tel que $a \in P$ et $b \notin P$.*

Démonstration. Par définition, $b \notin \uparrow a$. On pose $(F \subset \mathcal{F}(L), \subseteq)$ la séquence pour l'inclusion des filtres contenant a. Comme la relation sous-jacente est bien fondée, on postule l'existence d'une borne supérieure, le filtre S. Il s'agit de démontrer maintenant la primalité de S. Posons $s, t \in S$. On admet que S n'est pas premier, on a alors $x \vee y \in S$ où $x, y \notin S$ sont deux éléments quelconques. Les filtres $\uparrow(x \wedge s)$ et $\uparrow(y \wedge t)$ contiennent alors b tels que :

$$x \wedge s \leq b$$
$$y \wedge t \leq b$$

ce qui implique $(x \wedge s) \vee (y \wedge t) \leq b$. Si on distribue maintenant le $(\vee)$ sur le $(\wedge)$, on a l'inégalité suivante :

$$(x \vee y) \wedge (x \vee t) \wedge (s \vee y) \wedge (s \vee t) \leq b$$

Trivialement $s \vee t \in S$. Puis, par définition de S, tous les suprema de s, t sont dans S donc, $(x \vee t), (s \vee y) \in S$. Il reste $x \vee y \leq b$, or cela réfute la preuve précédente de $b \notin S$. Donc l'hypothèse est fausse et on a :

$$x \vee y \in S \Rightarrow x \in S \text{ ou } y \in S$$

S est donc un filtre premier. □

Grâce à ce lemme, les propriétés de maximalité et de primalité coincident, à la fois, pour les filtres et pour les idéaux. Celles-ci permettent donc de définir une décomposition unique et non redondante par le biais de l'ensemble $\mathcal{J}(L)$ d'un treillis L dont les filtres sont par conséquent premiers.

Théorème 3.3.2. *Soit un treillis distributif $(L, \wedge, \vee)$, les propriétés suivantes sont établies :*

1. *L est isomorphe au treillis des idéaux de l'ordre ;*

2. *Chaque élément $a \in L$ possède une décomposition unique en $a = \bigwedge_{J \subseteq \mathcal{J}(L)} J$;*

Démonstration. Étant donné que les filtres sont des parties fermées pour l'opérateur $(\wedge)$, l'intersection des filtres contenant un élément x est encore un filtre. On en déduit que l'idéal engendré par x atteint précisément les atomes engendrant ces filtres. On définit la carte suivante :

$$\phi : L \to \mathcal{P}(\mathcal{J}(L)) \tag{3.130}$$
$$x \mapsto \downarrow x \cap \mathcal{J}(L) \tag{3.131}$$

Par la première équation du théorème (3.3.1), on sait déjà que : $\phi(x \wedge y) = \phi(x) \cap \phi(y)$. Prouver l'isomorphisme nécessite de démontrer que $\phi(.)$ commute sur l'opérateur $(\vee)$. Par définition, on a : $\phi(x) \cup \phi(y) \subseteq \phi(x \vee y)$. Étant donné un élément $j \in \phi(x \vee y)$, on a alors :

$$\begin{aligned} & j \leq x \vee y \\ \iff & j = j \wedge (x \vee y) \\ \iff & j = (j \wedge x) \vee (j \wedge y) \end{aligned}$$

Puisque $j \in \phi(x \vee y)$, alors j est irréductible et donc $j \in \phi(x) \cup \phi(y)$. □

Le corollaire de ce théorème est que les éléments d'un treillis distributif admettent une représentation unique, linéaire sur la dimension de l'univers et indépendante des autres éléments du treillis par le lemme de séparation.

3.4 Conclusion

Dans ce chapitre, nous avons rappelé les divers résultats et méthodes de construction des ensembles ordonnés et des treillis, plus particulièrement. En particulier, nous avons réalisé la passerelle entre la manipulation des congruences d'un système logique au sein d'un treillis par la voie du théorème de représentation de Birkhoff pour les algèbres booléennes.

Nous allons maintenant prolonger cette approche au cadre où les couples filtre-idéal sont substitués par les couples formés du contexte associé à une requête d'agrégation, c.-à-d. ses paramètres formels, et les modèles instanciés sous la forme de partitions d'ensemble. En particulier, nous verrons que le rapport entre chaque instance suit le même schéma involutif : la dualité de l'ordre s'exprime entre la structure associée aux requêtes et la structure d'ordre des partitions, c.-à-d. plus le contexte est contraint, moins le nombre de classes est élevé.

4

Agrégation dans les entrepôts de données

Dans ce chapitre, nous exposons notre première proposition en vue d'opérationnaliser les partitions d'ensemble et leur calcul lorsque celles-ci modélisent des résultats de requêtes d'agrégation sur des données multidimensionnelles. Ce travail s'appuie en partie sur les résultats présentés dans [44].

4.1 Introduction

Les outils d'analyse de données multidimensionnelles tels que les environnements permettant de faire du *Traitement analytique en ligne*[1] sont conçus pour procurer une interface conviviale permettant de construire, faire la maintenance et interroger des données agrégées, suivant plusieurs axes de recherche ou dimensions.

Ces outils visent également à fournir des représentations pertinentes de l'information ainsi collectée. En outre, ils permettent la production de rapports de synthèse offrant une vue transversale sur les activités d'une entreprise par une instance décisionnaire et l'évaluation de la robustesse de futures analyses par l'emploi de diagrammes de toutes sortes.

En règle générale, un utilisateur est directement confronté à la métaphore conceptuelle imposée par le système afin de communiquer avec, et disposer des données qu'il

[1] en anglais, On-Line Analytical Processing

contient de manière productive. Les dialectes SQL OLAP et MDX s'appuient tous les deux sur une extension du modèle relationnel de Codd (R-OLAP) permettant d'exprimer des requêtes sur des ensembles de données agrégées.

Cette souplesse permet en outre la rétrocompatibilité avec les données relationnelles, et permet de calculer les réponses à des requêtes OLAP par le biais des moteurs d'évaluation SQL habituels. La caractéristique essentielle de ces outils est alors l'infrastructure mise à disposition pour gérer les requêtes sur des agrégats de manière transparente et permettre leur combinaison.

Transactions				
#Id	Produit	Client	Zone	Ventes
1	p_1	c_1	z_2	10
2	p_1	c_3	z_1	20
3	p_2	c_2	z_1	30
4	p_2	c_3	z_3	10
5	p_3	c_3	z_3	20

TABLE 4.1 : Une table des faits, présentant un ensemble de ventes

Dans le but d'illustrer les principales idées de ce chapitre, nous introduisons dans la Table (4.1) une base de données jouet qui reporte des unités de vente selon un schéma produit-client-zone (`P-C-Z`). Un analyste est supposé construire de nombreux résumés à partir de données brutes afin de fournir des conseils utiles dans le but de

1. dresser un inventaire pour chaque point de vente ;
2. annuler une ligne de produit qui n'est pas une activité viable ;
3. accroître la rentabilité par la fermeture des magasins les moins rentables ;

et ainsi de suite.

Chaque critère individuel est appelé à fonder la base d'une décision sur laquelle s'appuiera une requête dédiée pourvoyant une vue résumée des informations collectées. La table offre un panorama détaillé des faits suivant les attributs `P-C-Z` et couplés à la mesure des `Ventes`.

En outre, la dimension `Zone` admet de l'information auxiliaire comme une hiérarchie de classification qui procure différents niveaux de granularité (*p.ex.* Ville ($\mathtt{Z}_2 = \mathtt{Z}$) $\rightarrow$ Région ($\mathtt{Z}_1$) $\rightarrow$ Pays ($\mathtt{Z}_0$)) afin de décrire les faits.

Parmi les données brutes, les faits sont donnés au grain le plus fin de sorte que les zones z_i référencent des villes. Se déplacer d'un niveau vers un autre aura pour but d'ajuster la finesse de l'analyse.

Réaliser une analyse en ligne nécessite principalement d'engendrer l'ensemble du *cube* de données de manière à dévoiler tous les agrégats, suivant toutes les perspectives imaginables en projetant selon toutes les combinaisons de dimensions.

Par exemple, les faits de la Table (4.1) produisent 15 *cuboïdes* (ou vues) que sont les projections selon P, C, $\mathtt{Z}_0$, $\mathtt{Z}_1$, $\mathtt{Z}_2$, PC, $\mathtt{PZ}_0$, $\mathtt{PZ}_1$, $\mathtt{PZ}_2$, $\mathtt{CZ}_0$, $\mathtt{CZ}_1$, $\mathtt{CZ}_2$, $\mathtt{PCZ}_0$, $\mathtt{PCZ}_1$, $\mathtt{PCZ}_2$ dans le but de matérialiser le cube de données. En particulier, la projection suivant les attributs $\mathtt{PCZ}_2$ est la table de faits, elle-même.

Cette énumération est généralement permise par l'emploi de vues précalculées, entreposant les agrégats de tuples afin de doper l'évaluation d'une requête à la volée. Puisque le nombre de vues croît exponentiellement avec le nombre de dimensions du cube, la matérialisation complète est un problème insurmontable pour des bases de données concrètes, même possédant un nombre modéré de dimensions. En particulier, seuls quelques sous-ensembles de vues peuvent être maintenus efficacement tandis que les autres doivent être calculés à la demande. Sélectionner les vues à matérialiser est un point crucial qui est traité dans ce chapitre.

De nombreux travaux suivent cet axe de recherche, la problématique étant d'extraire depuis un cube de données $\mathcal{C}$ un sous-ensemble de cellules qui satisfont une certaine propriété ϕ. Une telle propriété ϕ est le plus généralement une formule conjonctive exprimée sur les attributs dimensionnels et les attributs de mesure. Cela peut être par exemple, les cellules ayant « 20 unités de vente » , ou « le client est c_3 et la zone est z_3 ».

La structure de ce résultat forme alors une classe d'équivalence sur les cellules $[.] \subseteq \mathcal{C} \times \mathcal{C}$ et telle que les assertions suivantes soient équivalentes :

$$\begin{array}{rl} & c_i, c_j \in \sigma_\phi(\mathcal{C}) \\ \iff & c_i \equiv_\phi c_j \\ \iff & [c_i]_\phi = [c_j]_\phi \end{array}$$

La première équation s'interprète par le fait que deux cellules satisfont toutes deux à la formule ϕ – exprimée dans un dialecte quelconque. Par conséquent, les deux formules qui suivent indiquent que les cellules impliquées sont donc *équivalentes* sous ce prisme et forment donc un ensemble représenté par ϕ.

Le problème est alors de *matérialiser* uniquement un petit ensemble de vues qui *subsume* toute la classe sous une relation prédéfinie $\leq$. Présumant que toutes les vues minimales dans un cube apportent une structure de treillis, alors n'importe quelle vue doit pouvoir être construite par la combinaison de celles-ci. Ces vues forment donc une couverture maximale des cellules, c'est-à-dire, elles représentent le plus petit ensemble de conditions par lesquelles les cellules forment une relation d'équivalence. Si ϕ est la formule associée à une vue minimale, alors pour toute formule ψ telle que $\phi \implies \psi$, les assertions suivantes sont équivalentes :

$$\begin{array}{rl} & \phi \leq \psi \\ \iff & [.]_\phi \supseteq [.]_\psi \end{array}$$

Lorsque $\varnothing \leq \phi$ est vraie – autrement dit ϕ est un axiome –, on établit alors la canonicité de ϕ et donc ϕ est la condition minimale pour laquelle les cellules $c_i \in [.]_\phi$ sont *maximalement* égales. La seconde équation est la conséquence directe : plus on ajoute de critères sélectifs, plus les contraintes sur les cellules sont prononcées, et par conséquent, moins de cellules seront mutuellement égales. Les vues minimales sont donc les modèles maximaux.

Ceci soulève donc plusieurs questions dont la plus évidente est : quelles sont les meilleurs représentations et méthodes de calcul de tels modèles ?

4.1.1 Notre proposition

Dans ce chapitre, nous ouvrons la voie vers un système de gestion d'agrégats se plaçant au-dessus d'un SGBD. Sur la base de motivations algébriques, nous proposons de modéliser les agrégats par des partitions définies sur des ensembles de tuples Ω.

Dans ce but, nous allons construire la structure $\Pi_\Omega \times \mathcal{P}(\mathcal{A})$ dont les couples mettent en relation les résultats de requêtes exprimées sur des sous-ensembles d'atttributs $X \subseteq \mathcal{A}$.

Suivant ce cadre, nous décrivons un modèle de données basique qui permet une recherche rapide de ses propriétés propres, notamment par l'étiquettage explicite des partitions par leur formule associée, et que nous appellerons *partitions annotées*.

De plus, étant donnée une table de faits stockée dans une base de données relationnelle, notre approche est capable de construire de manière incrémentale, un ensemble de partitions canoniques qui capte suffisamment d'informations du cube réduisant à peau de chagrin le coût de construction. Par la suite, on évalue les requêtes d'agrégation par rapport aux partitions d'ensembles.

La suite de ce chapitre suit la progression suivante. Nous présenterons dans la section 2 des travaux relatifs aux différentes représentations des cubes de données et à l'usage des relations d'équivalence dans ce contexte. Dans la section 3, nous présenterons notre modèle de données pour partitioner la base de données sous-jacente d'un entrepôt de données et illustrerons comment retrouver les informations originales sur les tuples. Dans la section 4, nous présenterons le treillis des partitions annotées et leurs propriétés. Celles-ci nous permettront de caractériser les vues. Nous procéderons ensuite à des expériences sur le jeu de données TPC-H dans la section 5, puis nous conclurons dans la dernière section.

4.2 Travaux connexes

De nombreuses propositions permettant la construction et la manipulation de vues précalculées d'un cube de données ont déjà été présentées. Les techniques utilisées pour la construction de tels résumés visent à préserver les mesures associées à chaque cellule tout en minimisant le nombre de vues nécessaire à leur interrogation.

Dans [81], Lakshmanan, Pei et Han réalisent un résumé global de la sémantique d'un cube de données, nommé le *cube quotient*, par la construction de classes. Chaque classe figure un ensemble de cellules satisfaisant à la même valeur mesurée, et qui sont donc regroupées ensemble. Dans certains cas, il est possible d'exhiber un chemin de

navigation le long des cellules de la classe, préservant la cohérence de l'analyse et par la suite des tendances intéressantes. Plus récemment, Brahmi et Ben Yahia [23] adaptent des concepts qu'on retrouve en fouille de règles pour interroger à l'aide de contraintes plus générales sur les cellules.

Dans un registre similaire, Casali *et al.* [27] ont introduit une nouvelle approche latticielle permettant l'organisation de résumés d'un cube de données, indépendante de la fonction d'agrégation, et dont l'efficacité ne repose pas sur la distribution des mesures pour chaque cellule. Cette méthode maintient des vues dont le contenu est un ensemble de cuboides dont la mesure associée est la même.

Comme cela a été établi précédemment, la combinatoire des cellules croît largement avec le nombre de dimensions, tandis qu'un grand nombre de nouvelles valeurs doit être calculé. Recherchant une représentation exacte, le maintien d'un bon équilibre entre le nombre de cellules dans le cube et le coût de traitement ne permet pas nécessairement de réduire de manière significative le nombre de vues à maintenir, même pour des mesures simples. De plus, il est peu vraisemblable que les approches « orientées mesure » soient en mesure de résumer efficacement une classe, étant donnée que la classe associée n'est généralement pas – algébriquement – close. Dans [123], Sismanis et Roussopoulos proposent des algorithmes permettant une représentation compressée exacte d'un cube de données et dont la taille croît polynomialement avec le nombre de dimensions.

En outre, étant donnée une requête ϕ, le calcul d'une requête similaire ψ telle que :

$$[.]_\phi = [.]_\psi$$

dans un cube de données est généralement considéré comme une fonctionnalité de haut-niveau offerte à l'utilisateur dans les environnements OLAP. Une telle fonctionnalité permet d'exhiber des parties du cubes qui sont reliées entre elles, généralement par l'entremise d'un module de réécriture de requêtes algébriques.

L'idée principale est de conserver la trace de l'exploration pendant que celle-ci est menée, puis de résumer l'histoire globale de manière cohérente et à la fin, fournir une assistance à l'utilisateur face à un grand nombre d'axes d'exploration [100, 117, 121]. Ce genre d'approche fonctionne de manière semblable à un système à recommandations qui conseille un utilisateur avec des astuces extraites depuis des expériences passées. Cependant, puisque les entrepôts de données sont « vivants » les changements appliqués au schéma conceptuel ou l'ajout de nouveaux faits dans la base de données sont susceptibles de rendre obsolète l'ancienne structure des résumés par rapport à une nouvelle version du cube de données.

De même, il est largement connu que l'usage des *dépendances fonctionnelles* (FDs) permet de réaliser le maintien de la cohérence d'une base de données tout le long de son cycle de vie. Munir un schéma relationnel de ces dépendances permet alors de déclencher des techniques de réécriture de requêtes utiles pour promouvoir ou déclasser l'application de clauses `GROUP BY` dans l'arbre associé à l'évaluation d'une expression algébrique. Dans ce but, Gupta *et al.* [63] décrivent un nouvel opérateur d'agrégation

généralisé muni de nouvelles règles de transformations améliorant le traitement de requêtes d'agrégation dans des requêtes complexes et permettant d'exploiter des vues matérialisées sur des résultats de requêtes d'agrégation.

En contrepartie, puisqu'un entrepôt de données évolue au gré du temps, essayer de surcontraindre la conception du schéma risque de produire l'effet inverse et entraver son utilisation. En effet, si on se représente son évolution par le biais d'une série temporelle, alors la différence entre les résumés produits risque d'augmenter. Garantir la pertinence des résumés attire ainsi l'attention de certains chercheurs, tentant de repérer des motifs couvrant une partie des faits sous certaines propriétés spécifiques et nommés *dépendances fonctionnelles conditionnelles* (CFDs) [19, 30, 37]. Cet outil permet de construire des motifs pertinents sur les données, c'est-à-dire des règles d'association (ARs), soit des dépendances qui tiennent uniquement sur une partie des tuples de la table.

Dans ce contexte, Garnaud *et al.* ont récemment proposé [57] de mettre à profit les propriétés latticielles émergeant naturellement des dépendances fonctionnelles et des axiomes d'Armstrong. En effet, étant donné deux ensembles d'attributs X, Y dans le schéma d'une relation, il est connu que la dépendance $X \rightarrow Y$ est satisfaite dès que $Y \subseteq X$. La structure obtenue est alors similaire au treillis des parties sur les attributs du schéma et par le biais de sa couverture, les auteurs montrent que des résultats de certaines requêtes d'agrégation peuvent être mutualisés de sorte à réduire le nombre de vues à maintenir, sans toutefois mettre en évidence que les requêtes associées forment alors des classes d'équivalence. Dans cette perspective, on mettra en évidence la proposition de [99] qui modélise également à l'aide de partitions d'ensemble les résumés associés à des vues matérialisées à l'aide de leur *treillis des concepts en accord*.

D'un point de vue plus prosaïque, les entrepôts de données étant mis en œuvre au-dessus d'un SGBD relationnel, les contraintes de rendement s'appliquent également. En sus des techniques d'indexation et de vues matérialisées, le partitionnement horizontal et vertical des relations apporte un surcroît de performance en influançant l'organisation physique des données [120, 4, 59, 73]. Cependant, le comportement de ces algorithmes dépend des facteurs de charge ou de l'usage de la base de données et n'aborde pas la partition comme un outil conceptuel de représentation, associée aux relations. Néanmoins, ces techniques sont également utilisées dans le but de produire des représentations compressées de relations, limiter la redondance des données et influencer les temps de traitement des requêtes [9].

La prochaine section sera dévolue à la modélisation d'un cube de données à partir de partitions d'ensembles.

4.3 Modélisation du cube de données

Le rôle de cette section est de présenter les principales définitions des concepts et outils permettant d'exploiter un cube de données dans un environnement relationnel.

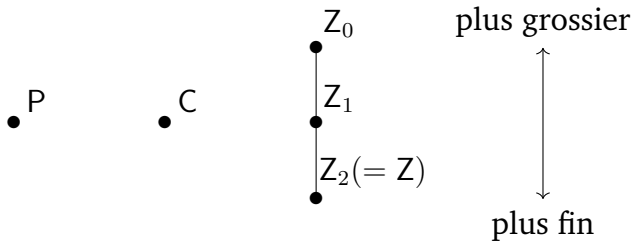


FIGURE 4.1 : Les trois dimensions, avec parallèlement les échelles de chaque attribut de la table des faits

4.3.1 Préliminaires

Un agrégat est principalement le résultat d'une requête `GROUP BY` paramétrée avec une liste d'attributs. Suivant l'exemple de la Table (4.1), les attributs sont définis parmi l'ensemble {`Produit`, `Client`, `Zone`} (abrégés par la suite P, C, Z). La clause `SELECT` intègre la mesure (`Ventes` dans l'exemple) construite depuis `count`, `avg`, `min` ou `max`, soient les fonctions d'agrégation les plus évidentes.

Étant donné un schéma relationnel $R : \mathcal{A}$ et une fonction d'agrégation $f : \mathcal{P}(R) \to \mathbb{R}^n$ qui fournit n mesures à un sous-ensemble de tuples dans R; on présume que les données brutes `r` sont un multiensemble de tuples dans R. Par exemple, chaque fait individuel sur `P-C-Z` y est consigné. Dans ce contexte, la table de faits r, aussi appelée le cuboide brut est simplement la contrepartie ensembliste de `r`, obtenue par l'élimination des doublons, puis étendue avec n nouveaux attributs, où les valeurs sont le résultat de l'application de f sur chaque sous-ensemble de dupliquets de `r`.

Par exemple, il y a dix transactions (p_1, c_1, z_2) qui ont produit la première ligne de la Table (4.1) avec f :=`count`.

Puisque notre principale préoccupation dépend de la partie structurelle du cube, nous utilisons un raccourci pour la définition complète du domaine d'un cuboide, à l'instar de l'ensemble $\mathcal{A}$ des attributs, et tel que :

$$r \subseteq \prod_{A \in \mathcal{A}} \mathsf{adom}(A)$$

sera abrégé en $r \subseteq \mathcal{A}$, omettant la mesure associée sans perte de généralité.

La Figure (4.1) montre tous les attributs et en particulier, l'échelle d'attributs Zone ($\mathsf{Z}_2 \to \mathsf{Z}_1 \to \mathsf{Z}_0$) du cube de l'exemple. Un cuboide est alors un sous-ensemble de cellules depuis n'importe quelle combinaison des attributs, sélectionnant au plus un attribut par dimension. De plus, les mots tuple, fait et cellule couvriront la même signification par la suite. Enfin, afin de faciliter la lecture, cube sera un terme générique pour cuboide lorsqu'il n'y a pas de risque d'ambiguïté.

Par exemple, $(\mathsf{P}, \mathsf{C}, \mathsf{Z}_0)$, (P, C) ou $(\mathsf{C}, \mathsf{Z}_1)$. Chaque fois qu'une dimension X n'est pas utilisée dans le cube, c'est équivalent à établir un taquet vers le haut pour chaque attri-

but où $X_\top$ est donné tel que $(\mathsf{C}, \mathsf{Z}_1) := (\mathsf{P}_\top, \mathsf{C}, \mathsf{Z}_1)$. Chaque $X_\top$ possède donc une valeur unique et neutre, disons $\mathsf{adom}(X_\top) = \{\star\}$.

Lors de l'exploration du cube dans $(\mathsf{P}, \mathsf{C}, \mathsf{Z}_i)$, les opérations d'exploration du cube vont présenter le comportement suivant :

- roll-up$(r, \mathsf{Z}_i) \subseteq (\mathsf{P}, \mathsf{C}, \mathsf{Z}_{i-1})$
- drill-down$(r, \mathsf{Z}_i) \subseteq (\mathsf{P}, \mathsf{C}, \mathsf{Z}_{i+1})$

Forer le long d'une dimension, par exemple depuis Z_i vers Z_{i+1}, ne peut être réalisé sans avoir au préalable réalisé une opération de synthétisation vers la dimension Z_{i+1} sachant que le niveau le plus fin est celui du cuboide brut et également le point de départ.

De plus, atteindre le niveau de détails le plus grossier d'une dimension est équivalent à réaliser une *projection* sur les autres dimensions. En effet, considérons le cube sur $(\mathsf{P}, \mathsf{C}, \mathsf{Z}_1)$; synthétiser selon P mène à rendre indiscernables les cellules selon cette dimension, alors le cube est défini sur $(\mathsf{P}_\top, \mathsf{C}, \mathsf{Z}_1)$, et donc $r \subseteq (\mathsf{C}, \mathsf{Z}_1)$.

Par la suite, considérons la requête d'agrégation q_P :

```
SELECT Produit, Sum(Ventes) as S
FROM Transactions
GROUP BY Produit
```

Cette requête résume la quantité globale de ventes par produit. Tous les tuples sont alors regroupés selon leur identifiant de produit et cela construit une relation d'équivalence sur les tuples de q_P telle que chaque classe soit définie par la formule suivante :

$$[t]_\mathsf{P} = \{u \mid u[\mathsf{P}] = t[\mathsf{P}]\}$$

et produit l'équivalence suivante :

$$(t, u) \in [.]_\mathsf{P} \iff t[\mathsf{P}] = u[\mathsf{P}]$$

Chaque ensemble dans $\mathcal{P}(\mathcal{A})$ permet de construire une relation binaire sur les tuples $\mathcal{P}(\Omega \times \Omega)$.

À la fin, il suffit d'observer que l'union des classes d'équivalence forme une couverture de tous les tuples de la relation r : $\bigcup_{t \in r}[t]_\mathsf{P} = \Omega$. Celle-ci dépend des attributs de regroupement X, c'est-à-dire l'attribut `Produit` dans notre exemple q_P.

La Table (4.2) fait la liaison implicite entre les faits bruts, possédant le même identifiant de produit et pouvant être résumé à l'aide d'une *partition* sur l'ensemble d'identifiants de tuples (*cf.* Table 4.1) :

$$q_\mathsf{P} = 12|34|5 \Leftrightarrow (12|34|5, \mathsf{P})$$

q_{P}	
Produit	Ventes
p_1	30
p_2	40
p_3	20

TABLE 4.2 : Ensemble des agrégats résultants de q_{P}

où « | » délimite chaque classe.

Les requêtes d'agrégation peuvent être vues comme un ensemble de projections où chaque classe est représentée par les tuples qui deviennent égaux par l'application d'une restriction sur les attributs projectifs, et finalement induit une partition définie sur ceux-ci.

Par exemple, les cartes suivantes donnent les règles explicites à partir desquelles le couple $(1|2|3|45, \{\mathsf{C}, \mathsf{Z}\})$ est réalisé.

$$\bigcup \left\{ \begin{array}{l} (c_1, z_2) \mapsto \{1\}, \\ (c_3, z_1) \mapsto \{2\}, \\ (c_2, z_1) \mapsto \{3\}, \\ (c_3, z_3) \mapsto \{4, 5\}, \end{array} \right\} \rightarrow (\{\{1\}, \{2\}, \{3\}, \{4, 5\}\}, \{\mathsf{C}, \mathsf{Z}\})$$

en utilisant une notation purement ensembliste. Comme le montre l'exemple ci-dessus, représenter le résultat d'une requête d'agrégation sous la forme d'une partition des tuples permet de recouvrer aisément les valeurs du domaine actif depuis les identifiants de tuples, en fonction de la classe à laquelle ils appartiennent.

L'ensemble des identifiants de tuples constitue par la suite l'*univers de définition* de r, et qui sera noté $\Omega \subset \mathbb{N}$.

Définition 4.3.1. *Étant donnés un sous-ensemble d'attributs X et une partition P figurant le modèle de la requête q_X, on nommera par la suite* partition annotée *le couple (P, X).*

$(\{\{1\}, \{2\}, \{3\}, \{4, 5\}\}, \{\mathsf{C}, \mathsf{Z}\})$ est alors une partition annotée.

Chaque fait ou tuple dans r est défini de manière unique sur le domaine de tous les attributs au degré le plus fin et constitue une clé composée sur la table. Regrouper alors sur *tous* les attributs construira seulement des singletons. Par conséquent, $q_{\mathsf{PCA}} := \bot$ où $\bot$ représente $1|2|3|4|5$.

Réciproquement, nous écrirons $\top$ au lieu de la partition 12345 pour résumer la requête d'agrégation $q_\varnothing$ qui ne réalise aucune distinction parmi les tuples, comme introduit dans la Table (4.3).

Propriété 4.3.1. *Étant donné un sous-ensemble d'attributs X, la projection sur X induit que les identifiants de tuples fusionnent, alors pour tout $Y \subseteq X$, on a :*

$$\forall t \in r, \quad [t]_X \supseteq [t]_Y$$

et détermine une dépendance fonctionnelle triviale $X \rightarrow Y$.

Carte entre les structures		
Tuples	Attributs	Partitions
distinct	P, C, Z	$\bot = \{\{\omega\} \mid \omega \in \Omega\}$
indiscernable	$\varnothing$	$\top = \{\Omega\}$

TABLE 4.3 : Borne inférieure et supérieure sur r suivant trois points de vue : tuples, attributs et partition.

Exploitant le fait que l'ensemble des dépendances fonctionnelles est en correspondance avec le treillis des attributs $\mathcal{A}$, à l'instar de [57], nous allons utiliser cette relation qui va nous permettre de relier la relation de raffinement entre deux partitions en fonction de la relation entre leur annotation respective :

$$Y \subseteq X \iff P \geq Q$$

avec les partitions annotées (P, X) et (Q, Y).

En outre, l'échelle de chaque dimension, par exemple $\mathsf{Z}_2 \to \mathsf{Z}_1 \to \mathsf{Z}_0$, représente un axe d'analyse navigable. Il définit également une dépendance sur le domaine actif associé à chaque niveau de granularité et transporte la structure hiérarchique, reliant les niveaux entre eux. Il est connu par [60, 78] qu'il est suffisant de collecter soit les niveaux les plus fins soit les plus grossiers de chaque dimension dans l'espace des attributs. Dans cette configuration, on ne réalise plus la distinction entre les différents niveaux de granularité et chacun d'entre eux est alors représenté comme un attribut indépendant.

4.3.2 Projections canoniques sur l'ensemble des attributs $\mathcal{A}$

Dans cette section, nous allons montrer que les classes de tuples provenant de requêtes d'agrégation canonique (sur un seul attribut) permettent d'exprimer des requêtes composées par le biais d'opérations ensembliste sur les classes de tuples tandis que le regroupement induit sur les valeurs du domaine actif reste cohérent. Puis, nous montrerons que ces opérations se généralisent au niveau des partitions elles-mêmes où l'annotation associée fournit le contexte.

Considérons par exemple, la Table (4.4) où les clients et les points de vente sont regroupés pour chaque identifiant de produit. Dans chaque colonne, figure les valeurs de son domaine actif regroupées selon le critère d'agrégation P. En outre, chaque valeur mesurée n'est pas mise à jour délibérément pour marquer le fait que son calcul peut être différé.

Le modèle associé à une requête d'agrégation q_X est donc la partition réalisée suivant les identifiants de tuples. Les Tables (4.5 et 4.6) représentent ensuite les modèles associés aux requêtes q_{C} et q_{Z}.

Comme le montre la dernière ligne des deux vues imbriquées q_{C} et q_{Z}, le client c_3 a acheté deux différents produits dans la zone z_3. La représentation multiensembliste

q_{P}				
Id's	P	C	Z	V
$\{1,2\}$	p_1	$\{\!\{c_1, c_3\}\!\}$	$\{\!\{z_1, z_2\}\!\}$	$+(10,20)$
$\{3,4\}$	p_2	$\{\!\{c_2, c_3\}\!\}$	$\{\!\{z_1, z_3\}\!\}$	$+(10,30)$
$\{5\}$	p_3	$\{\!\{c_3\}\!\}$	$\{\!\{z_3\}\!\}$	20

TABLE 4.4 : Vue imbriquée des tuples de $\mathcal{A}\backslash\{\mathsf{P}\}$

q_{C}				
Id's	C	P	Z	V
$\{1\}$	c_1	$\{\!\{p_1\}\!\}$	$\{\!\{z_2\}\!\}$	10
$\{3\}$	c_2	$\{\!\{p_2\}\!\}$	$\{\!\{z_1\}\!\}$	30
$\{2,4,5\}$	c_3	$\{\!\{p_1, p_2, p_3\}\!\}$	$\{\!\{z_1, z_3, z_3\}\!\}$	$+(20,+(10,20))$

TABLE 4.5 : Vue imbriquée des tuples de q_{C}

$\{\!\{z_1, z_3, z_3\}\!\}$ permet de préserver à la fois la taille des classes construites à partir des identifiants uniquement, mais aussi la répartition des valeurs d'attribut dans chaque classe. Il est alors possible de scinder $\{\!\{z_1, z_3, z_3\}\!\}$ en $\{\!\{z_1\}\!\}$ et $\{\!\{z_3, z_3\}\!\}$, selon la sous-partition $2|45$, ce qui mène à leur vue imbriquée $q_{\mathsf{C,Z}}$ (*c.f.* Table 4.7). En parcourant les combinaisons restantes sur les attributs, il apparaît aucune autre dépendance sur les valeurs du domaine actif permettant de regrouper d'autres tuples au sein d'une même classe.

La Table (4.8) exhibe chaque partition reliée à chaque requête d'agrégation. On voit immédiatement qu'il existe plusieurs requêtes qui peuvent être résumées suivant la *même* partition d'ensemble tandis que les attributs X fournissent le contexte approprié.

À la lecture des différentes partitions modélisant tous les résultats de requêtes d'agrégation, il apparaît qu'une seule partition est en mesure de résumer plusieurs requêtes à la fois. Cela induit donc de nouvelles classes, définies cette fois-ci sur les attributs eux-mêmes. Nous allons être en mesure de construire des classes $[.] \subseteq \mathcal{P}(\mathcal{A}) \times \mathcal{P}(\mathcal{A})$ telles que :

$$\forall (P, X), (Q, Y) \in \Pi^{\Omega} \times \mathcal{P}(\mathcal{A}), Y \in [X] \Rightarrow P = Q$$

où (P, X) et (Q, Y) sont les modèles respectifs des requêtes q_X et q_Y (*cf.* la définition 4.3.1). Le but est d'éviter les redondances permettant de conserver un ensemble de partitions définies sur *les tuples*, puis d'autre part, sur l'espace *des attributs* et ne conserver uniquement que les partitions liées à une requête particulière.

Dans ce qui suit, nous allons montrer comment construire la relation entre les requêtes d'agrégation sur différentes sélections d'attributs et leur modèle afin de ne retenir que l'information pertinente.

q_{Z}				
Id's	Z	P	C	V
$\{2,3\}$	z_1	$\{\!\{p_1,p_2\}\!\}$	$\{\!\{c_3,c_2\}\!\}$	$+(20,30)$
$\{1\}$	z_2	$\{\!\{p_1\}\!\}$	$\{\!\{c_1\}\!\}$	10
$\{4,5\}$	z_3	$\{\!\{p_2,p_3\}\!\}$	$\{\!\{c_3,c_3\}\!\}$	$+(10,20)$

TABLE 4.6 : Vue imbriquée des tuples de q_{Z}

$q_{\mathsf{C,Z}}$			
Id's	C × Z	P	V
$\{1\}$	(c_1,z_2)	$\{\!\{p_1\}\!\}$	10
$\{2\}$	(c_2,z_1)	$\{\!\{p_1\}\!\}$	20
$\{3\}$	(c_3,z_1)	$\{\!\{p_2\}\!\}$	30
$\{4,5\}$	(c_3,z_3)	$\{\!\{p_2,p_3\}\!\}$	$+(10,20)$

TABLE 4.7 : Vue imbriquée des tuples de $q_{\mathsf{C,Z}}$

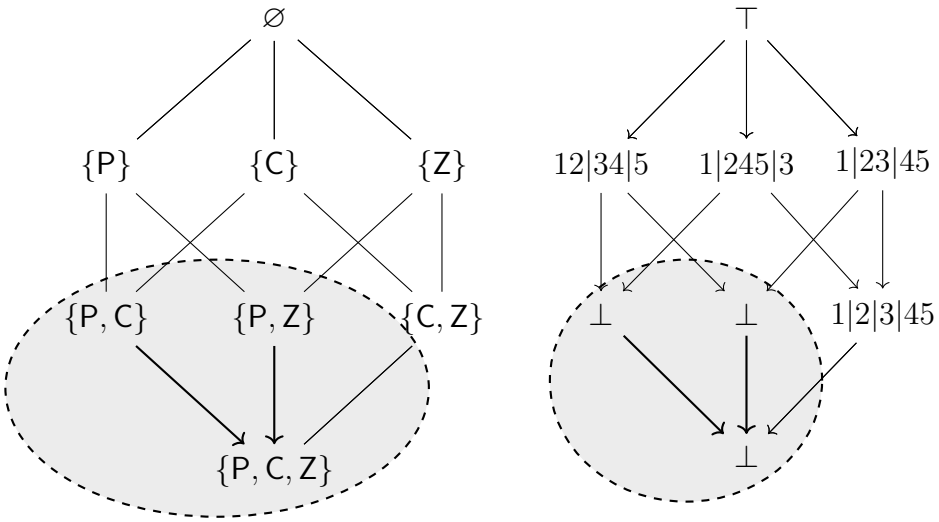


FIGURE 4.2 : Treillis des parties $X \subseteq \mathcal{A}$ et leurs modèles $P \in \Pi^{\Omega}$

4.4 Construction du treillis des partitions annotées

La figure (4.2) représente le treillis des sous-ensembles d'attributs $(\mathcal{P}(\mathcal{A}), \subseteq)$. Pour chaque sous-ensemble d'attributs X, il associe donc la partition correspondante comme résultat de la requête q_X.

Néanmoins, la structure représentée n'est pas celle du treillis des partitions. Comme indiqué auparavant, certaines combinaisons d'attributs ont pour modèle la même partition, ce qui va nous mener à construire une relation entre l'algèbre des partitions et l'algèbre des attributs. La figure (4.3) représente le treillis de l'ordre associé aux partitions annotées de $\Pi^{\Omega} \times \mathcal{P}(\mathcal{A})$, que l'on note $\Pi^{\Omega}_{\mathcal{A}}$.

Dans un premier temps, intéressons-nous à la fonction qui associe une partition des

$X \in \mathcal{P}(\mathcal{A})$	$P \in \Pi^{\Omega}$
$q_{\varnothing}$	$\top$
q_{P}	$12\|34\|5$
q_{C}	$1\|3\|245$
q_{Z}	$1\|23\|45$
$q_{\mathsf{P},\mathsf{Z}}$	$\bot$
$q_{\mathsf{C},\mathsf{Z}}$	$1\|2\|3\|45$
$q_{\mathsf{P},\mathsf{C}}$	$\bot$
$q_{\mathsf{P},\mathsf{C},\mathsf{Z}}$	$\bot$

TABLE 4.8 : Attributs des requêtes d'agrégation *vs.* les partitions associées

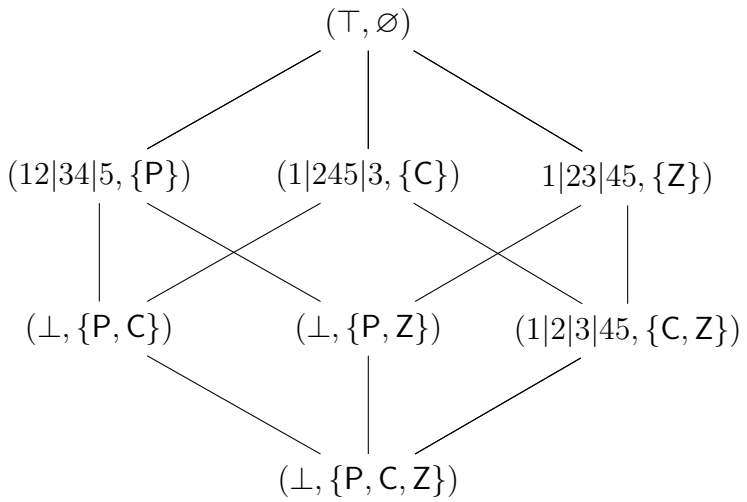


FIGURE 4.3 : Treillis des partitions annotées $\Pi^{\Omega}_{\mathcal{A}}$

tuples à un sous-ensemble d'attributs :

$$\begin{aligned} f : \mathcal{P}(\mathcal{A}) &\to \Pi^{\Omega} \\ X &\mapsto \{[t]_X \mid t \in r\} \end{aligned}$$

avec $f(\varnothing) = \top$ and $f(\mathcal{A}) = \bot$. Cette fonction est une définition formelle de ce que réalise une requête d'agrégation sur les identifiants de tuples Ω. Nous allons maintenant mettre en évidence que les relations d'équivalence associées à chaque partition, sont également des *congruences* et donc stables par opération.

On remarquera que pour toute partition annotée $(P, X) \in \Pi^{\Omega}_{\mathcal{A}}$, on a $P = f(X)$. Ce qui signifie que la structure $\Pi^{\Omega}_{\mathcal{A}}$ représente l'ensemble des modèles maximaux d'agrégation des tuples associées à chaque annotation.

Théorème 4.4.1. *Étant donnée une collection de sous-ensembles* $\{X_i\} \in \mathcal{P}(\mathcal{P}(\mathcal{A}))$ *:*

$$f(\bigcup_i X_i) = \bigwedge_i f(X_i) \tag{4.1}$$

Démonstration. Soit les tuples $x, y \in \Omega$, on suppose que $[x] = [y] \in f(\bigcup_i X_i)$, on a alors :

$$\forall A \in \bigcup_i X_i, x[A] = y[A] \tag{4.2}$$
$$\iff \forall i \forall A \in X_i, x[A] = y[A] \tag{4.3}$$
$$\iff \forall i, [x] \in f(X_i) \tag{4.4}$$
$$\iff [x] \in \bigwedge_i f(X_i) \tag{4.5}$$

□

Corollaire 4.4.1. *Le morphisme f est antitone, c.-à-d. , pour tout $X, Y \in \mathcal{P}(\mathcal{A})$ tels que $X \subseteq Y$, on a $f(Y) \leq f(X)$.*

Démonstration. Étant donné $X \subseteq Y$, on a alors :

$$f(Y) = f(\mathcal{A} \cap Y) \tag{4.6}$$
$$= f(X \cup X^c \cap Y) \qquad (X \cup X^c = \mathcal{A}) \tag{4.7}$$
$$= f(X) \wedge f(X^c \cap Y) \tag{4.8}$$
$$\leq f(X) \tag{4.9}$$

□

Ce corollaire met en évidence le caractère involutif entre les requêtes d'agrégation et leur modèle : plus le nombre de critères d'agrégation augmente, plus le nombre de tuples en accord diminue.

En particulier, prenons un sous-ensemble d'attributs X, la partition résultante est alors calculable par la formule suivante :

$$f(X) = \bigwedge_{A \in X} f(A)$$

Le tableau (4.8) est une bonne illustration de cette propriété. Prenons les partitions associées aux attributs {C} et {Z} :

$$\begin{aligned} f(\{\mathsf{C}, \mathsf{Z}\}) &= f(\{\mathsf{C}\}) \wedge f(\{\mathsf{Z}\}) \\ &= 1|3|245 \wedge 1|23|45 \\ &= 1|2|3|45 \end{aligned}$$

Ce qui est bien la partition associée à $\{\mathsf{C}, \mathsf{Z}\}$ et mène à la prochaine définition.

Définition 4.4.1. *Soit deux partitions annotées (P, X) et (Q, Y) de $\Pi^{\Omega}_{\mathcal{A}}$, on définit l'opérateur $(\sqcap)$ par la formule suivante :*

$$(P, X) \sqcap (Q, Y) = (P \wedge Q, X \cup Y) \tag{4.10}$$

On remarque que la formule duale $f(X \cap Y) = f(X) \vee f(Y)$ n'est pas définie partout et ne permet pas de définir $(\sqcup)$ sur les partitions annotées de façon duale, c.-à-d. par $(P \vee Q, X \cap Y)$. Par exemple,

$$f(\{\mathsf{C},\mathsf{Z}\}) \vee f(\{\mathsf{P},\mathsf{Z}\}) = 1|2|3|45 < f(\{\mathsf{Z}\}) = f(\{\mathsf{C},\mathsf{Z}\} \cap \{\mathsf{P},\mathsf{Z}\})$$

En effet, l'application du $(\vee)$ nécessite le maintien de la fermeture transitive entre les classes de tuples, ce qui n'est pas garanti par les partitions résultantes d'un $(\wedge)$. Nous verrons donc par la suite dans quelle condition le $(\vee)$ est applicable et les répercussions pratiques que cela engendre.

En dépit de la capacité à calculer des partitions annotées par l'utilisation de l'opérateur $(\wedge)$, calculer la fonction inverse, qui cherche un sous-ensemble d'attributs correspondant est de loin plus intéressant.

En effet, cela va nous permettre de mettre en évidence une relation d'équivalence menant à la construction d'une famille d'ensembles d'attributs $\mathcal{P}(\mathcal{A})$. Sur la figure (4.2), on observe que chaque partition est associée à plusieurs sous-ensembles d'attributs, donc nous devrions être capable de mettre en évidence des éléments remarquables et reflétant la structure induite par une classe d'équivalence.

Définition 4.4.2. *Soit deux partitions annotées (P, X) et (Q, Y) de $\Pi^{\Omega}_{\mathcal{A}}$, on définit la relation d'inclusion $(\sqsubseteq)$ par la formule suivante :*

$$(P, X) \sqsubseteq (Q, Y) \tag{4.11}$$
$$\iff P \leq Q \text{ et } X \supseteq Y \tag{4.12}$$
$$\iff (P, X) \sqcap (Q, Y) = (P, X) \tag{4.13}$$

Tout d'abord, on définit la fonction qui associe la partition P au plus grand sous-ensemble d'attributs possible :

$$g : \Pi^{\Omega} \to \mathcal{P}(\mathcal{A}) \tag{4.14}$$
$$P \mapsto \bigcup \{X \in \mathcal{P}(\mathcal{A}) \mid \exists [t] \in P, [t] \subseteq [t]_X\} \tag{4.15}$$

Il est important de noter que le morphisme g admet également une définition indirecte par l'usage de f. En effet, étant donné que f inverse tous les éléments de $\mathcal{P}(\mathcal{A})$, la définition suivante est admissible (*c.f.* à partir du théorème 3.2.2) :

$$f^c : \Pi^{\Omega} \to \mathcal{P}(\mathcal{A}) \tag{4.16}$$
$$P \mapsto \bigcup \{X \in \mathcal{P}(\mathcal{A}) \mid P \leq f(X)\} \tag{4.17}$$

et est semblable à réaliser l'union des antécédents X dont l'application par f est un majorant de P.

Posons par exemple, les partitions $P = 1|2|3|45$ et $Q = \bot$, il est aisé de voir que les deux définitions sont équivalentes :

$$g(P) = f^c(P) = \bigcup \{\varnothing, \{\mathsf{C}\}, \{\mathsf{Z}\}\}$$

puisque $P \leq f(\{C\})$ et $P \leq f(\{Z\})$, donc on a : $f^c(P) = \{\mathsf{C}, \mathsf{Z}\}$. Concernant Q, on a :

$$f^c(Q) = \bigcup\{X \in \mathcal{P}(\mathcal{P}(\mathcal{A}))\} = \{\mathsf{P}, \mathsf{C}, \mathsf{Z}\}$$

Théorème 4.4.2. *Étant donnée une collection de partitions $\{P_i\} \in \mathcal{P}(\Pi^\Omega)$:*

$$f^c(\bigwedge_i P_i) = \bigcup_i f^c(P_i) \tag{4.18}$$

Démonstration. Soit un ensemble $A \in \mathcal{A}$, on suppose que $A \in f^c(\bigwedge_i P_i)$, on a alors :

$$\bigwedge_i P_i \leq f(A) \tag{4.19}$$
$$\iff \forall P \in \{P_i\}, P \leq f(A) \tag{4.20}$$
$$\iff \forall i, A \in f^c(P_i) \tag{4.21}$$
$$\iff A \in \bigcup_i f^c(P_i) \tag{4.22}$$

□

Les morphismes (f, f^c) forment donc un couple involutif entre les attributs de requêtes et leur modèle. Nous verrons plus loin que leur composition permet d'établir des propriétés importantes sur les partitions annotées.

Définition 4.4.3 (Représentation des partitions annotées)**.** *Soit $X \in \mathcal{P}(\mathcal{A})$, le morphisme $\phi^{\mathcal{A}}$ construit la partition annotée correspondante à la requête d'agrégation q_X :*

$$\phi^{\mathcal{A}} : \mathcal{P}(\mathcal{A}) \to \Pi^\Omega_{\mathcal{A}} \tag{4.23}$$
$$X \mapsto (f(X), X) \tag{4.24}$$

Étant donnée une partition $P \in \mathcal{P}(\mathcal{A})$, le morphisme ϕ^Π renvoie les paramètres de la requête dont le résultat est le majorant le plus proche de P, au sens de f^c :

$$\phi^\Pi : \Pi^\Omega \to \Pi^\Omega \times \mathcal{P}(\mathcal{A}) \tag{4.25}$$
$$P \mapsto (P, f^c(P)) \tag{4.26}$$

On remarque ici que les ensembles d'arrivée diffèrent pour $\phi^{\mathcal{A}}$ et ϕ^Π, étant donné que $\phi^{\mathcal{A}}$ construit nécessairement le modèle figurant le résultat d'une requête. Inversement, ϕ^Π associe les paramètres d'une requête dont le résultat est similaire mais pas nécessairement égal, et le résultat peut donc ne pas être une partition annotée.

Un corollaire trivial est que $\mathcal{P}(\mathcal{A})$ est isomorphe à $\Pi^\Omega_{\mathcal{A}}$ et $\phi^{\mathcal{A}}$ est l'injection canonique qui préserve l'identité de chaque sous-ensemble d'attributs lorsqu'on lui adjoint sa partition.

Depuis cette définition, on obtient une version plus générale du théorème (4.4.1).

Théorème 4.4.3. *Soit une collection d'ensembles* $\{X_i\} \in \mathcal{P}(\mathcal{P}(\mathcal{A}))$*, on a l'équivalence suivante :*

$$\phi^{\mathcal{A}}(\bigcup_i X_i) = \bigsqcap_i \phi^{\mathcal{A}}(X_i) \tag{4.27}$$

Démonstration. En utilisant la définition (4.4.1), on passe facilement d'un côté à l'autre de l'expression :

$$\phi^{\mathcal{A}}(\bigcup_i X_i) = (f(\bigcup_i X_i), \bigcup_i X_i) \tag{4.28}$$

$$= (\bigwedge_i f(X_i), \bigcup_i X_i) \tag{4.29}$$

$$= \bigsqcap_i (f(X_i), X_i) \tag{4.30}$$

□

La figure (4.4) schématise la relation fonctionnelle entre l'ensemble des partitions annotées et ses projections canoniques. Le théorème suivant émerge naturellement.

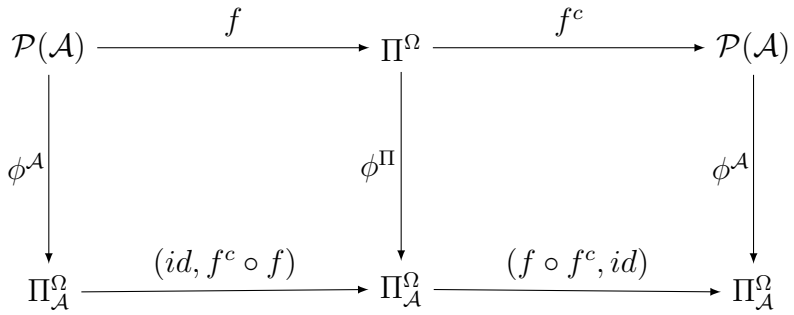


FIGURE 4.4 : Diagramme de composition des morphismes f^c et f dans $\Pi^{\Omega}_{\mathcal{A}}$

Théorème 4.4.4. *Étant donnée une partition annotée* $(P, X) \in \Pi^{\Omega}_{\mathcal{A}}$*, les propositions suivantes sont équivalentes :*

1. $\phi^{\mathcal{A}}(X) = (\phi^{\Pi} \circ f)(X)$ *;*
2. X *est un point fixe de l'opérateur* $f^c \circ f$ *;*

Démonstration. En partant de la définition de $\phi^{\mathcal{A}}(X) = (f(X), X)$, en substituant P par $f(X)$ dans la définition de ϕ^{Π} : $\phi^{\Pi}(f(X)) = (f(X), (f^c \circ f)(X))$, d'où l'équivalence suivante :

$$X = (f^c \circ f)X \tag{4.31}$$

$$\iff (f(X), X) = (f(X), (f^c \circ f)(X)) \tag{4.32}$$

$$\iff \phi^{\mathcal{A}}(X) = (\phi^{\Pi} \circ f)(X) \tag{4.33}$$

□

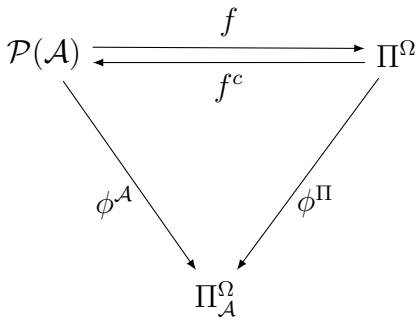


FIGURE 4.5 : Diagramme de composition des morphismes lorsque $f^c \circ f$ est le morphisme identité

En revanche, ϕ^{Π} construit toutes les requêtes possibles sur Π^{Ω} par l'entremise de f^c qui nécessite l'application d'une fermeture sur $\mathcal{P}(\mathcal{A})$; plusieurs requêtes ayant le même modèle. On en déduit que $\phi^{\mathcal{A}}$ induit la même structure si elle est appliquée uniquement sur la fermeture de ces ensembles d'attributs. La figure (4.5) symbolise cette simplification.

Par exemple, la figure (4.3) représente la structure induite par $\phi^{\mathcal{A}}$ tandis que la figure (4.6) est celle construite selon l'application $\phi^{\Pi} \circ f$. La différence entre les deux structures est donc représentée par les partitions annotées vérifiant la propriété suivante.

Définition 4.4.4 ($\mathcal{A}$-maximalité)**.** *Soit une partition annotée $(P, X) \in \Pi^{\Omega}_{\mathcal{A}}$, celle-ci est dite $\mathcal{A}$-maximale si et seulement si pour toute partition annotée (Q, Y) :*

$$P = Q \implies Y \subseteq X$$

Il est aisé de voir sur la figure (4.6) que les partitions annotées engendrées par l'application de $\phi^{\Pi} \circ f$ sont toutes $\mathcal{A}$-maximales. Par la suite, on utilisera la notation $P_X \triangleq (P, X) \in \Pi^{\Omega}_{\mathcal{A}}$ pour désigner les partitions annotées.

Comme nous l'avons vu plus haut, l'isomorphisme exprime l'idée que toute requête q_X sur un sous-ensemble d'attributs X mène à une partition. En revanche, une partition quelconque n'est pas nécessairement le modèle d'une requête. Par exemple, la partition $1|24|3|5$ n'est pas le modèle d'une requête, cependant la requête q_C exprime un résultat similaire à celle-ci, de sorte que $\phi^{\Pi}(1|24|3|5) = (1|24|3|5, \{C\})$.

Théorème 4.4.5. *Étant donnée une partition annotée $P_X \in \Pi^{\Omega}_{\mathcal{A}}$, les propositions suivantes sont équivalentes :*

1. $\phi^{\Pi}(P) = (\phi^{\mathcal{A}} \circ f^c)(P)$;
2. *P est un point fixe de l'opérateur $f \circ f^c$;*

Ce théorème s'obtient de manière duale au théorème (4.4.4) et la preuve est par conséquent similaire, en tout point. En particulier, l'étude de la composition $f \circ f^c$ est

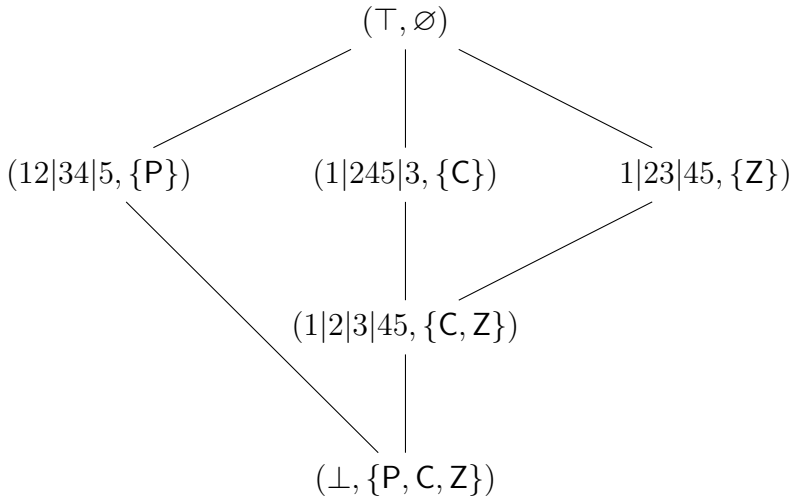


FIGURE 4.6 : Treillis des partitions annotées $\Pi^{\Omega}_{\mathcal{A}}$ selon $\phi^{\Pi} \circ f$

moins intéressante que la précédente. En effet, la construction de chaque modèle associé à une requête réalise implicitement un point fixe sur $\mathcal{P}(\mathcal{P}(\Omega))$ où, parmi la collection induite, chaque sous-ensemble de tuples est déjà pris maximalement.

La composition des morphismes f et f^c permet donc d'obtenir deux autres morphismes aux propriétés intéressantes. Le premier admet la définition suivante :

$$\begin{aligned} f^c \circ f : \mathcal{P}(\mathcal{A}) &\to \mathcal{P}(\mathcal{A}) \\ X &\mapsto \bigvee \{Y \in \mathcal{P}(\mathcal{A}) \mid f(X) = f(Y)\} \end{aligned}$$

Il constitue alors une fermeture algébrique qui renvoie à tout sous-ensemble X, le plus grand sur-ensemble $Y \supseteq X$ tel que $\phi^{\mathcal{A}}$ soit une partition annotée $\mathcal{A}$-maximale.

Réciproquement, on a le morphisme :

$$\begin{aligned} f \circ f^c : \Pi^{\Omega} &\to \Pi^{\Omega} \\ P &\mapsto \bigwedge \{Q \in \Pi^{\Omega} \mid f^c(P) = f^c(Q)\} \end{aligned}$$

Prouvons maintenant que c'est une opération de fermeture sur $\mathcal{P}(\Pi^{\Omega})$ qui renvoie la partition annotée la plus proche dans le treillis $\Pi^{\Omega}_{\mathcal{A}}$.

Théorème 4.4.6. *Soit une partition P, alors on a la propriété suivante :*

$$P \leq (f \circ f^c)(P) \tag{4.34}$$

Démonstration. La preuve s'obtient aisément en décomposant suivant f^c :

$$f^c(P) = \bigvee \{X \mid P \leq f(X)\} \tag{4.35}$$

$f^c(P) = f^c(Q)$ implique donc qu'il existe une famille de partitions $\{R_i\}$ où $P, Q \leq f(X)$ telle que $f(X) \leq \bigwedge_i R_i$ et :

$$P, Q \leq \bigwedge_i R_i \tag{4.36}$$

$$\implies P \leq (f \circ f^c)(P) \tag{4.37}$$

□

En l'occurrence, on a :

$$(f^c \circ f)(\{\mathsf{P}, \mathsf{C}\}) = f^c(\bot) = \{\mathsf{P}, \mathsf{C}, \mathsf{Z}\}$$

tandis que :

$$(f \circ f^c)(1|24|3|5) = f(\mathsf{C}) = 1|245|3$$

On déduit alors de ces deux exemples que ni $\{\mathsf{P}, \mathsf{C}\}$ ni $1|24|3|5$ ne sont des point-fixes et on a les résultats suivants.

Définition 4.4.5. *Le sous-ensemble d'attributs X est la requête maximale (au sens de l'inclusion) de sa classe d'équivalence si et seulement si $(f^c \circ f)(X) = X$.*

Chaque requête maximale dépend donc d'une partition annotée $\mathcal{A}$-maximale et constituera le représentant de sa classe.

Théorème 4.4.7. *Une partition P est un modèle maximal d'une requête d'agrégation si et seulement si P est un point-fixe de $f \circ f^c$ tel que $(f \circ f^c)(P) = P$.*

Démonstration. C'est le corollaire du théorème (4.4.6). Il suffit de voir que $P = (f \circ f^c)(P)$ implique que l'ensemble des antécédents de la fonction f est nécessairement unique. □

Cette approche va ainsi nous permettre de *quotienter* la structure algébrique induite en instanciant uniquement les partitions annotées $\mathcal{A}$-maximales. La structure induite constitue alors un système de générateurs de partitions annotées, nous permettant de recouvrer l'ensemble de la structure $\Pi^{\Omega}_{\mathcal{A}}$.

Définition 4.4.6 (Treillis des partitions annotées $\mathcal{A}$-maximales)**.** *Soit $(\mathcal{B}^{\Omega}_{\mathcal{A}}, \sqcap, \sqcup, \sqsubseteq, \bot, \top)$ le treillis borné engendré par $\Phi^{\Pi} \circ f$ tel que $\mathcal{B}^{\Omega}_{\mathcal{A}} \subseteq \Pi^{\Omega}_{\mathcal{A}}$. On utilise les notations suivantes :*

$$P_X \leq_{\Pi} Q_Y \iff P \leq Q \tag{4.38}$$
$$P_X \leq_{\mathcal{A}} Q_Y \iff X \supseteq Y \tag{4.39}$$

Étant données les partitions annotées $P_X, Q_Y \in \Pi^{\Omega}_{\mathcal{A}}$, on a les propriétés suivantes :

$$P_X =_{\mathcal{A}} Q_Y \implies P_X =_{\Pi} Q_Y \tag{4.40}$$
$$P_X =_{\Pi} Q_Y \implies P_X =_{\mathcal{A}} Q_Y \tag{4.41}$$

Enfin, les taquets sont définis suivant ceux de Π^{Ω} par $\bot \triangleq \bot_{\mathcal{A}}$ et $\top \triangleq \top_{\varnothing}$.

On définit maintenant les opérateurs associées.

Définition 4.4.7 (Opérateurs). *Étant données les partitions annotées $P_X, Q_Y \in \mathcal{B}^{\Omega}_{\mathcal{A}}$, les opérations suivantes sont bien définies :*

$$(P, X) \sqcap^{+} (Q, Y) = (P \wedge Y, (f^c \circ f)(X \cup Y)) \tag{4.42}$$
$$(P, X) \sqcup (Q, Y) = (P \vee Y, X \cap Y) \tag{4.43}$$

La structure $\mathcal{B}^{\Omega}_{\mathcal{A}}$ joue ici deux rôles. Son premier rôle est de fournir un sous-ensemble de partitions annotées qui soit une partie génératrice de $\Pi^{\Omega}_{\mathcal{A}}$. Le second est plus exotique, étant donné que la structure est également un treillis, la relation $(\sqsubseteq)$ permet de naviguer entre différents systèmes de générateurs. L'opérateur $(\sqcap^{+})$ réalise donc le calcul d'une borne[2] dans $\mathcal{B}^{\Omega}_{\mathcal{A}}$ tandis que $(\sqcap)$ réalise l'opération usuelle dans $\Pi^{\Omega}_{\mathcal{A}}$.

Par exemple, posons $P_X = (12|34|5, \{\mathsf{P}\})$ et $Q_Y = (1|245|3, \{\mathsf{C}\})$, on a :

$$\begin{aligned} P_X \sqcap Q_Y &= (\bot, \{\mathsf{P}, \mathsf{C}\}) \in \Pi^{\Omega}_{\mathcal{A}} \\ P_X \sqcap^{+} Q_Y &= (\bot, \{\mathsf{P}, \mathsf{C}, \mathsf{Z}\}) \in \mathcal{B}^{\Omega}_{\mathcal{A}} \end{aligned}$$

Il est important de noter que contrairement au cas général sur $\Pi^{\Omega}_{\mathcal{A}}$, l'opérateur $(\sqcup)$ est bien défini. Par exemple, ajoutons un attribut fictif F à $\mathcal{A}$ tel que $q_{\mathsf{F}} = 14|25|3$, la structure engendrée par $\phi^{\pi} \circ f$ admet une borne supérieure non-triviale – les opérandes sont distinctes du résultat – comme le montre la figure (4.7). En effet, on vérifie aisément que :

$$f(\{\mathsf{C}, \mathsf{Z}\}) \vee f(\{\mathsf{C}, \mathsf{F}\}) = 1|245|3 = f(\{\mathsf{C}\}) \tag{4.44}$$

On obtient ainsi quelques résultats intéressants sur cette structure.

Théorème 4.4.8. *$\mathcal{B}^{\Omega}_{\mathcal{A}}$ est le plus grand ensemble de partitions annotées devant être matérialisé.*

Revenons à la Figure (4.7), les partitions annotées devant être retenues sont donc :

$$\mathcal{B}^{\Omega}_{\{\mathsf{P},\mathsf{C},\mathsf{Z},\mathsf{F}\}} = \{P_{\mathsf{P}}, P_{\mathsf{C}}, P_{\mathsf{Z}}, P_{\mathsf{F}}, P_{\mathsf{C},\mathsf{Z}}, P_{\mathsf{C},\mathsf{F}}, P_{\mathsf{P},\mathsf{C},\mathsf{Z},\mathsf{F}}\}$$

Il est proprement inutile dans cet exemple de matérialiser la représentation des requêtes portant sur $\mathsf{PC}, \mathsf{PZ}, \mathsf{PF}, \mathsf{ZF}, \mathsf{PCZ}, \mathsf{PCF}, \mathsf{PZF}, \mathsf{CZF}$. Il s'agit donc bien d'une représentation « compressée » du cube.

Parmi les partitions annotées retenues, on distingue les partitions annotées $P_{\mathsf{C},\mathsf{F}}$, $P_{\mathsf{C},\mathsf{Z}}$ et $P_{\mathsf{P},\mathsf{C},\mathsf{Z},\mathsf{F}}$ qui peuvent être précalculées par $\bigsqcap_{A\in\{\mathsf{C},\mathsf{F}\}} P_A$, $\bigsqcap_{A\in\{\mathsf{C},\mathsf{Z}\}} P_A$ et $\bigsqcap_{A\in\{\mathsf{P},\mathsf{C},\mathsf{Z},\mathsf{F}\}} P_A$.

Les autres partitions, toutes correspondant au modèle d'une requête sur un seul attribut, ne peuvent donc être engendrées par une combinaison $(\sqcap)$ de partitions annotées.

Ces quatre partitions, $P_{\mathsf{P}}, P_{\mathsf{C}}, P_{\mathsf{Z}}$ et P_{F}, forment le plus petit ensemble de partitions annotées devant être matérialisé.

[2]La présence du $+$ suscrit souligne l'usage de la fermeture $\langle\cdot\rangle^{+} \equiv f^c \circ f$ permettant l'obtention du représentant de la classe d'équivalence dans $\mathcal{P}(\mathcal{A})$.

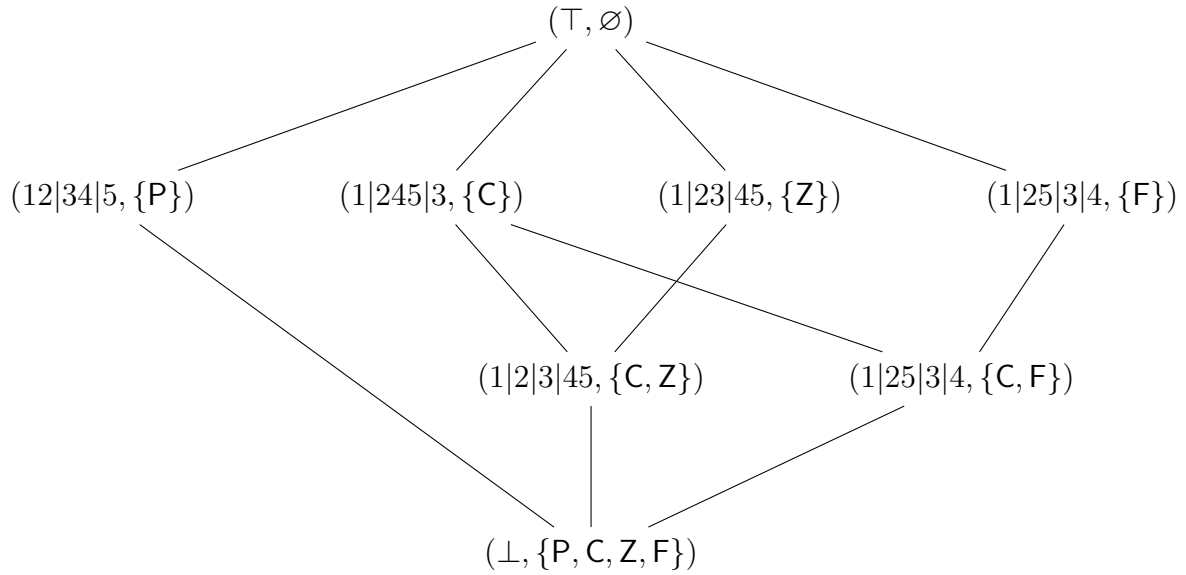


FIGURE 4.7 : Treillis des partitions annotées $\Pi^{\Omega}_{\mathcal{A}}$ avec $\mathcal{A} = \{\mathsf{P}, \mathsf{C}, \mathsf{Z}, \mathsf{F}\}$.

Définition 4.4.8 (Partition atomique)**.** *Soit une partition annotée $P_X \in \Pi^{\Omega}_{\mathcal{A}}$, elle est $\sqcap$-irréductible si et seulement si :*

$$\forall Q_Y, R_Z \in \Pi^{\Omega}_{\mathcal{A}}, P_X = Q_Y \sqcap R_Z \implies P_X = Q_Y \text{ ou } P_X = R_Z \tag{4.45}$$

Nous appelerons partitions atomiques*, les partitions $\sqcap$-irréductibles.*

Cette définition est également valable dans $\mathcal{B}^{\Omega}_{\mathcal{A}}$ pour l'opérateur $(\sqcap^{+})$. Les partitions atomiques sont alors déterminées par l'ensemble des partitions annotées obtenu pour chaque requête portant sur un seul attribut.

Théorème 4.4.9. *Soit l'ensemble des partitions atomiques $\mathcal{J}(\mathcal{B}^{\Omega}_{\mathcal{A}})$, on a le résultat suivant :*

$$\Pi^{\Omega}_{\mathcal{A}} = \bigsqcap \mathcal{J}(\mathcal{B}^{\Omega}_{\mathcal{A}}) \tag{4.46}$$

Puisque les partitions de $\mathcal{J}(\mathcal{B}^{\Omega}_{\mathcal{A}})$ sont déterminées par une requête sur un seul attribut, chaque combinaison sur $\mathcal{P}(\mathcal{A})$ engendre alors une partition annotée différente.

Par ailleurs, $P_{\mathsf{C}} = P_{\mathsf{C},\mathsf{Z}} \bigsqcup P_{\mathsf{C},\mathsf{F}}$ est calculée par l'application de $(\sqcup)$. Il est alors possible que certaines partitions atomiques puissent être substituées avec deux autres partitions annotées dont les annotations sont composées de plusieurs attributs, ce qui mène au résultat suivant.

Théorème 4.4.10. *Soit les sous-ensembles d'attributs $X, Y \in \mathcal{P}(\mathcal{A})$, alors la* dépendance multivaluée *(MVD) [50] $X \twoheadrightarrow Y$ équivaut à :*

$$P_X = P_{X \cup Y} \sqcup P_{X \cup \mathcal{A} \setminus Y} \tag{4.47}$$

Démonstration. $X \twoheadrightarrow Y$ induit la décomposition de la relation suivant le ou les attributs X avec :

$$r = r[XY] \bowtie r[XZ], \quad Z = \mathcal{A} \backslash Y$$

Les tuples dans chaque projection sont donc en relation, de chaque côté, avec toutes les valeurs du domaine actif de X tels que :

$$\forall (y, z) \in r[Y] \times r[Z], \exists x \in R[X], (x, y, z) \in r$$

Chaque x étant associé à une classe de P_X, pour tout $(x, y, z) \in r$, les classes $[y] \in P_{X \cup Y}$ et $[z] \in P_{X \cup Z}$ sont telles que $[x] = [y] \circ [z]$. On en déduit que chaque classe $[x] \in P_X$ est la fermeture transitive de $[y] \circ [z]$.

Réaliser $P_{X \cup Y} \vee P_{X \cup Z}$ calcule la fermeture transitive des classes se chevauchant dans chaque partition. Elle réalise donc la construction complète de la relation $(X \cup Y) \times (X \cup Z)$. Étant donné que la dépendance indique que la décomposition préserve la fermeture transitive des tuples, par conséquent le $(\sqcup)$ est apte à recouvrir la relation de départ. □

Un corollaire important est donc que lorsque l'opérateur $(\sqcup)$ est valide, alors les partitions atomiques P_X peuvent ne pas être également $\sqcup$-irréductibles. Un exemple détaillé sera exploité dans les analyses de nos résultats afin de mettre en relief cet aspect des partitions annotées.

Lorsqu'on souhaite calculer le résultat d'une requête d'agrégation q_X, deux choix s'offrent à nous :

- Étant donné un ensemble $X \in \mathcal{P}(\mathcal{A})$, calculer $\phi^{\mathcal{A}}(X)$ et afficher la partition résultante ;
- Calculer P telle que $(\phi^{\Pi} \circ f)(P)$ existe dans $\Pi^{\Omega}_{\mathcal{A}}$, si la partition annotée a été matérialisée, la retourner.

La structure de treillis de $\mathcal{B}^{\Omega}_{\mathcal{A}}$ laisse également entrevoir la possibilité que la relation entre les différentes partitions annotées nous permette de choisir plusieurs systèmes de générateurs autres que $\mathcal{J}(\mathcal{B}^{\Omega}_{\mathcal{A}})$. Dans ce cas, le treillis des partitions annotées $\Pi^{\Omega}_{\mathcal{A}}$ pourra être reconstruit depuis une $\bigsqcup\bigsqcap$-expression sur un sous-ensemble différent de $\mathcal{B}^{\Omega}_{\mathcal{A}}$.

Nous allons maintenant présenter nos résultats d'expériences appliquées à un jeu de données synthétiques.

4.5 Expériences

Afin de caractériser le coût de traitement induit par le calcul des résultats des requêtes sur des partitions, nous reportons ci-dessous les résultats expérimentaux réalisés à l'aide du jeu de données TPC-H[3] avec le facteur d'échelle établi à 1 ($6\,000\,000$ tuples). Pour être

[3] http ://www.tpc.org/tpch/default.asp

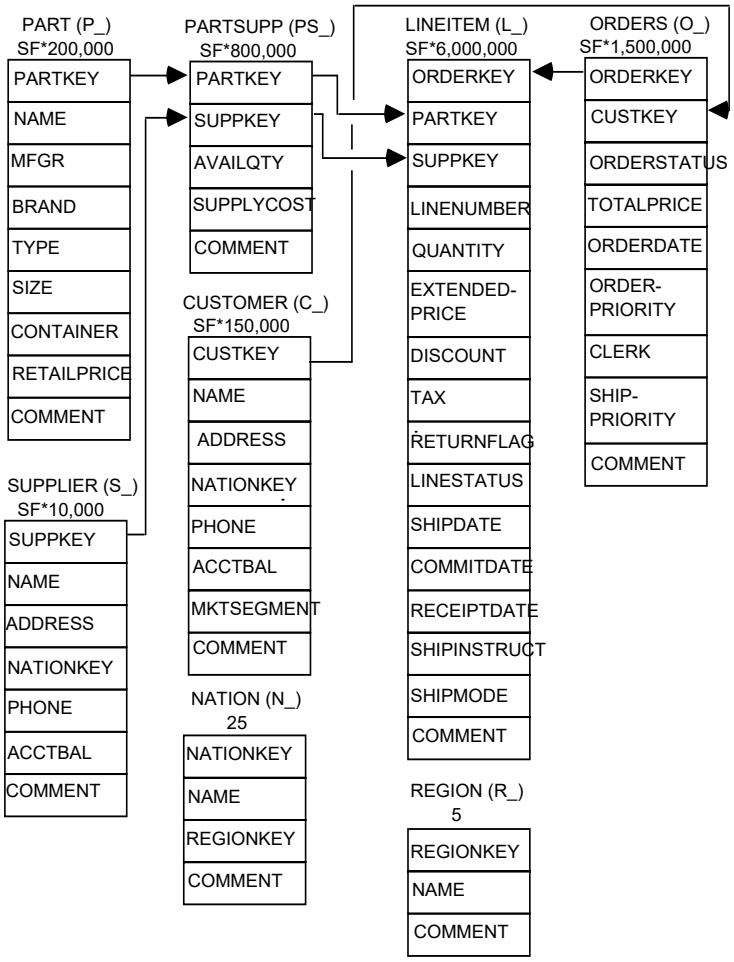


FIGURE 4.8 : Le schéma du jeu de données TPC-H

en mesure de construire des partitions sur les instances de la base de données associée, nous avons appliqué quelques légers changements aux schémas.

Dans une veine comparable aux auteurs de [101], nous visons à effectuer des transformations propres à nous permettre d'engendrer des partitions dont l'univers de définition est le même, quel que soit le contexte – les attributs – lui étant rattaché.

L'organisation de la base étant réalisée suivant un schéma en flocons de neige, nous allons réaliser une table unique en joignant successivement toutes les tables figurant les dimensions jusqu'à la table de faits.

Chaque ensemble de tables est joint suivant les chemins induits par les dépendances de clés étrangères du schéma. Par exemple, les tables `nation` et `region` sont jointes deux fois, la première fois avec `partsupp`, la seconde fois avec `orders`. Donc, ceci devrait éviter toute ambiguïté sur l'usage des attributs de localisation qu'ils soient destinés à décrire des fournisseurs ou des clients. Enfin, toutes les tables intermédiaires sont jointes avec la table de faits, `lineitem`.

Concernant le nombre de tuples de la table de faits, aucun changement n'a lieu et

par la suite, chaque partition sera définie sur un univers de tuples commun Ω, celui-ci énumérant $6\,000\,000$ tuples différents

La seule différence avec l'instance originale de la base réside dans le fait que l'ensemble des attributs – soit environ 70 – est disponible avec ses valeurs et il permet de réaliser des requêtes conjonctives sans devoir exprimer de jointure, ce qui fournit un net avantage. En contrepartie, les requêtes d'agrégation qui faisaient auparavant référence aux tables auxiliaires et leurs attributs nécessitent dorénavant d'examiner un plus grand nombre de tuples. Nous allons montrer par la suite que cela ne déteriore pas les performances.

4.5.1 Aspects computationnels

Mettre en œuvre de manière efficace notre solution dépend du modèle de données utilisé pour stocker en mémoire centrale[4] l'ensemble des partitions impliquées dans une requête et par la suite interroger selon les caractéristiques qui nous intéressent. Une partition étant trivialement un ensemble de sous-ensembles de Ω, nous avons essentiellement besoin de calculer des intersections ensemblistes entre classes, et appartenant à des opérandes différentes.

La représentation logique des partitions suit un modèle relationnel utilisant un système de représentants permettant de reproduire les opérations algébriques sur des partitions d'ensembles (*cf.* [45]). Chaque représentant est donc choisi de manière déterministe au sein de sa classe telle que la représentation en mémoire d'une partition soit un mot obtenu sur l'alphabet Ω et l'accès à l'identifiant de classe d'un élément se fasse en temps constant. Par exemple, la partition $P = 12|34|5$ aura pour mot 11335 tandis que $Q = 1|25|34$ s'écrira 12332.

Deux éléments appartiennent alors à la même classe dans une partition si ils partagent le même représentant. Traiter $P \wedge Q$ engendre un nouveau mot qui s'écrit 12335 et qui est donc une vue sérialisée de la partition résultante, matérialisant l'encodage de la relation d'appartenance sur Ω. On constate par exemple que 3 et 4 appartiennent à la classe $[3]$ dans $P \wedge Q$. Aucune méthode d'optimisation particulière n'est toutefois utilisée pour traiter une séquence n-aire $\bigwedge_{i=1}^{n} P_i$; le coût probable de l'application de toutes les opérations sera de $(n-1) \times O(P \wedge Q)$.

4.5.2 Processus opérationnel global

Les auteurs dans [20] réalisent une étude approfondie sur la pertinence des modèles de requêtes proposés par TPC-H, en fonction de l'application visée et des caractéristiques essentielles du système que l'on souhaite éprouver. Il apparaît que le partionnement

[4] La technologie « mémoire centrale distribuée » rend acceptable cette hypothèse à large échelle, même si avec $6 \cdot 10^6 \times 70$ données nous avons rencontré aucune difficulté, sans architecture spécifique à ce stade.

vertical d'un schéma d'une base de données n'est pas un concept nouveau et certains articles ont promu précédemment ce type d'approche (*cf.* [84]).

Néanmoins, nous revendiquons la légitimité de notre approche, celle-ci se fondant sur des propriétés moins invasives car complémentaires. En outre, l'instanciation de la structure que nous présentons à la fin de la précédente section est optimale en terme d'espace.

Notre but est donc de traduire des séquences de clauses `GROUP BY`, définies sur un sous-ensemble d'attributs du schéma $\mathcal{A}$, et produire l'ensemble des partitions P_{A_i} représentant les modèles associés aux requêtes pouvant être formulées suivant ces descripteurs. Le traitement est ensuite réalisé par le calcul de l'expression suivante :

$$\prod_{A \in X} P_A$$

Suivant le cadre exposé précédemment, nous transcrivons une séquence de clauses `GROUP BY` appliquée sur un sous-ensemble d'attributs $X \in \mathcal{P}(\mathcal{A})$ en une requête conjonctive sur les partitions associées aux attributs de X. Le traitement prend fin une fois réalisé l'ensemble des $(\sqcap)$ binaires menant à $\prod_{A \in X} P_A$. Dans son implémentation courante, le calcul de la requête ne propose pas d'optimisation en fonction de l'ordonnancement des $(\sqcap)$. Chaque classe du résultat figure alors un agrégat particulier, calculé sur le domaine actif de $\Pi_{A \in X} D_A$.

Pour mettre en application ce scénario, nous avons sélectionné une douzaine de requêtes intéressantes. Afin de résumer comment nous simulons une requête d'agrégation, nous allons jeter un coup d'œil à la requête n°3 qui renvoie les dix premiers tuples construits par :

```
SELECT
    l_orderkey, o_orderdate, o_ship_priority,
    sum(l_extendedprice*(1-l_discount)) as revenue,
FROM
    customer, orders, lineitem
WHERE
    c_mktsegment = 'AUTOMOBILE'
    and c_custkey = o_custkey
    and l_orderkey = o_orderkey
    and o_orderdate < date '1995-03-01'
    and l_shipdate > date '1995-03-01'
GROUP BY
    l_orderkey,o_orderdate,o_ship_priority
ORDER BY
    revenue desc, o_orderdate
```

Ayant à notre disposition une table nous évitant le calcul d'une jointure entre les trois tables `customer`, `orders` et `lineitem`, seul le traitement des tâches suivantes reste à réaliser :

1. calculer les tuples impliqués par le prédicat conjonctif :
   `c_mktsegment = 'AUTOMOBILE'`,
   `o_orderdate < '1995-03-01'`,
   `l_shipdate < '1995-03-01'`

2. depuis les attributs
   `l_orderkey`,
   `o_orderdate`,
   `o_ship_priority`
   , se munir des partitions

$$P_{\texttt{l\_orderkey}}, P_{\texttt{o\_orderdate}}, P_{\texttt{o\_ship\_prior}}$$

3. réaliser le tri décroissant des tuples suivant `revenue` et `o_orderdate`

La première opération nécessite uniquement d'appliquer une restriction à l'univers $\Omega' \subset \Omega$ sur lequel sont définies les partitions. Sachant que nous concentrons nos efforts sur l'application des opérateurs latticiels des partitions, pour réaliser cette opération, nous nous contenterons de réduire le nombre de tuples pris en considération.

Comme dans les moteurs de requêtes SQL, le filtrage des tuples est réalisé à la volée lors du parcours des partitions pour la construction du $(\sqcap)$: l'opération consiste à interroger un dictionnaire qui met en correspondance les représentants de chaque classe avec sa modalité sur l'attribut dans la relation. En conséquence, le surcoût engendré par la sélection est négligeable. La procédure de tri, quant à elle, en vertu du nombre faible de résultats attendus des requêtes TPC-H, peut également être négligée dans l'évaluation du coût.

Une fois obtenu le résultat de $P_{\texttt{l\_orderkey}} \wedge P_{\texttt{o\_orderdate}} \wedge P_{\texttt{o\_ship\_prior}}$, chaque classe caractérise alors une ligne de la table devant être affichée, la mesure suivant l'attribut numérique `revenue` est calculée en sommant l'ensemble des mesures de chaque tuple impliqué dans la classe.

Dans un second temps, nous allons également évaluer le nombre de partitions annotées devant être matérialisées. Cela nécessite le calcul de toutes les combinaisons d'attributs possibles, représentant donc 2^{69} expressions algébriques, ce qui est insurmontable dans des conditions opérationnelles. Nous avons donc procédé de manière itérative en construisant une base de partitions atomiques de manière incrémentale, et en mesurant à chaque étape le nombre de nouvelles partitions annotées représentant une nouvelle classe d'équivalence. Le nombre de partitions que nous allons utiliser pour constituer notre ensemble sera limité à une vingtaine d'entre elles.

Ce choix est éclairé, si on se réfère au schéma de la base, par le fait qu'une grande part des attributs sont des clés dont le domaine actif contient le plus grand nombre de valeurs, ce qui implique que le nombre de classes dans la partition associée est également très grand et donc la partition très proche de $\perp$. Une observation que l'on peut réitérer pour les attributs relatifs aux dates ou aux commentaires.

Les partitions atomiques, une vingtaine ayant servi de base pour ce test, sont parmi celles offrant le plus de possibilité de combinaisons et de diversité car choisies en fonction de leur *rang* : cette fonction mesure la position relative d'une partition par rapport à la partition singleton, $\perp = 1|2|3|4|\ldots$, dans le treillis des partitions. Elle est définie par la formule suivante :

$$\mathrm{rk}(P) = |\Omega| - |P|$$

On déduit aisément que $\mathrm{rk}(\perp) = 0$ et que la partition ne possédant qu'une seule classe est de rang maximal, soit $\mathrm{rk}(\top) = |\Omega| - 1$. De plus, étant données deux partitions P_1 et P_2, la fonction de rang est transitive : l'ordre est préservé après une opération. Présumons que $P_1 < P_2$ dont on déduit immédiatement que $\mathrm{rk}(P_1) < \mathrm{rk}(P_2)$ alors pour toute partition Q, on aura :

$$\mathrm{rk}(P_1) \leq \mathrm{rk}(P_2) \implies \mathrm{rk}(P_1 \wedge Q) \leq \mathrm{rk}(P_2 \wedge Q)$$

Cette identité a pour conséquence que le rang décroît après l'application de chaque $(\wedge)$ d'une expression. La probabilité d'atteindre la partition $\perp$ décroît dès lors que le rang des partitions choisies comme opérandes est accru. Afin d'obtenir un nombre de classes le plus réaliste possible – et se rapprochant du pire cas –, les partitions atomiques sont donc choisies en priorité parmi celles ayant le plus petit nombre de valeurs dans leur domaine actif, donc de rang maximal.

Afin d'illustrer notre propos, posons l'instance d'une base de données sur un schéma composé de trois attributs binaires comme le montre le Tableau (4.9) : Celle-ci recense

Ω	A	B	C
1	0	0	0
2	0	0	1
3	0	1	0
4	0	1	1
5	1	0	0
6	1	0	1
7	1	1	0
8	1	1	1

TABLE 4.9 : Une instance sur le schéma $\{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$

l'ensemble de tous les mots binaires sur 3 bits et est donc maximalement hétérogène car contenant tous les tuples possibles. L'ensemble des partitions atomiques ne contient alors que des partitions n'ayant que deux classes.

La figure (4.9) illustre parfaitement cette diversité et l'absence de partitions annotées redondantes, quelle que soit la requête calculée. Le choix de partitions binaires permet d'obtenir un nombre maximal de partitions annotées et appuie notre approche expérimentale afin de maximiser ce nombre. Incidemment, on remarque aussi que les attributs vérifient des MVDs, étant données les identités suivantes :

$$\begin{cases} P_{\mathsf{A}} = P_{\mathsf{A,B}} \sqcup P_{\mathsf{A,C}} \\ P_{\mathsf{B}} = P_{\mathsf{A,B}} \sqcup P_{\mathsf{B,C}} \\ P_{\mathsf{C}} = P_{\mathsf{A,C}} \sqcup P_{\mathsf{B,C}} \end{cases}$$

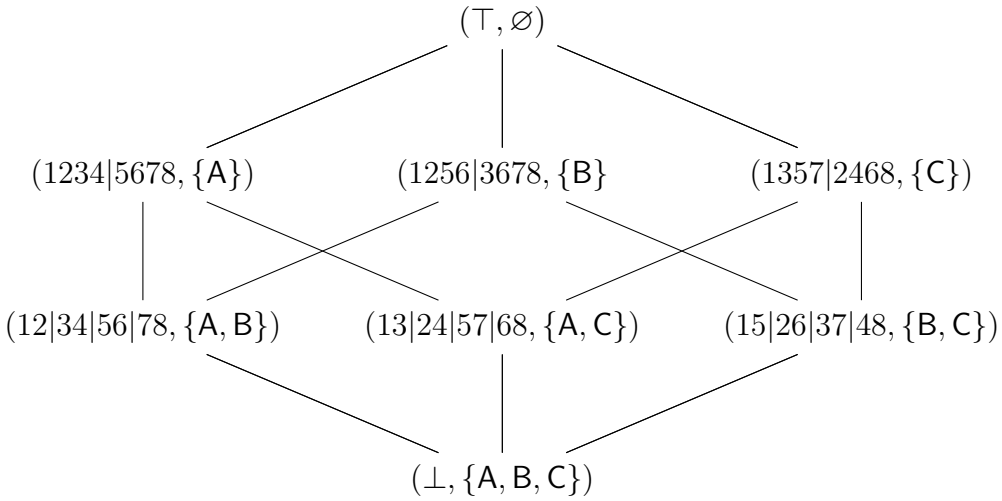


FIGURE 4.9 : Instances du treillis $\mathcal{B}^{\Omega}_{\{\mathsf{A},\mathsf{B},\mathsf{C}\}}$

et permettent l'émergence d'une dépendance de jointure comme le montre la Figure (4.10). Cela montre que le critère généralement retenu pour la décomposition de re-

Ω	A	B
12	0	0
34	0	1
56	1	0
78	1	1

(a)

$\bowtie$

Ω	A	C
13	0	0
24	0	1
57	1	0
68	1	1

(b)

FIGURE 4.10 : Décomposition de la relation en $\bowtie$ (AB, AC)

lations, c.-à-d. l'existence d'une MVD/JD – permettant le partitionnement vertical, est typiquement orthogonal au nôtre (*cf.* [51, 10, 3]). En effet, on serait plutôt enclin à souligner le fait que les partitions $P_{\mathsf{A}}, P_{\mathsf{B}}, P_{\mathsf{C}}$ peuvent ne pas être matérialisées car directement expressibles par les partitions P_{AB}, P_{AC}, P_{BC}. Néanmoins, les remplacer ne présente pas un grand intérêt car dans le cas d'un schéma $\mathcal{A}$ tel que $|\mathcal{A}| = m > 3$, le nombre de partitions annotées à deux attributs sera dans ce contexte de $\binom{m}{2}$, qui est très largement supérieur à m.

Toutes les expérimentations ont été faites sur un ordinateur doté d'un processeur intel Core i7-3610@2.3Ghz et tournant sous Ubuntu 14.04. La mise en œuvre des opérations a été réalisée en C++ (compilé sous `g++ -O3`) ainsi que la structure de données utilisée pour la représentation des instances associées à chaque partition. Chaque exécution et sa mesure associée a été menée dans un contexte mono-processus. Les résultats sont présentés pour l'ensemble des requêtes sélectionnées dans le tableau (4.10). Dans tous les cas de figure, le temps consommé pour calculer chaque expression algébrique simulant la requête semble raisonnable. Il apparaît assez clairement que le temps de traitement est directement fonction du nombre de partitions à utiliser et donc du nombre

#requête	partitions	temps (ms)
1	2	235
2	2	180
3	3	420
7	3	400
9	3	490
10	7	1150

TABLE 4.10 : Performances globales des requêtes (la colonne du milieu fait référence au nombre de critères du `GROUP BY` traitées pour chaque requête)

d'attributs de regroupement de la requête originale, comme l'atteste la requête n°10. Ce fait devrait suggérer l'usage des dépendances fonctionnelles $X \rightarrow Y$ définies au niveau du schéma afin d'éviter le calcul d'une expression $P_X \sqcap P_Y = P_X$ qui est inutile comme également indiqué dans [20].

Bien que la mise en œuvre de nos opérations et le schéma opérationnel mériteraient certaines améliorations, les résultats que nous présentons sont acceptables, même comparés à ceux publiés dans [93], dans la mesure où le passage à un autre facteur d'échelle aura un impact linéaire sur le temps de traitement des requêtes. En effet, le temps nécessaire à la mise en œuvre d'un $(\sqcap)$ est linéairement dépendant du nombre de tuples à visiter lorsqu'on balaie chaque partition.

Concernant la possibilité d'éviter les calculs conjonctifs entre des partitions, en recherchant la classe d'équivalence associée à un ensemble d'attributs X, nous reportons quelques résultats additionnels sur le niveau de compression applicable aux données de TPC-H.

En notant A_n, la séquence représentant la cardinalité de chaque ensemble, on constate sur la Figure (4.11) que le nombre de partitions annotées C_n est proche de n^2 avec $C_n = o(n^2)$.

En tenant compte de tous les attributs du schéma, cela nécessiterait en extrapolant, le stockage d'environ $2\,000$ partitions. Bien que cela représente un effort beaucoup plus important que le stockage de la base elle-même, on rappelle que le stockage de tous les attributs ne présente pas nécessairement d'intérêt pour le processus exploratoire, réduisant du même coup, la matérialisation des partitions associées à certaines requêtes.

Dans l'absolu, il est peu vraisemblable qu'il soit nécessaire également de matérialiser l'ensemble des $o(n^2)$ partitions annotées représentant chaque classe d'équivalence, puisque comme le montre la structure $\mathcal{B}^{\Omega}_{\mathcal{A}}$, il existe certainement de nombreuses redondances entre elles permettant de réduire ce nombre à un plus petit sous-ensemble et d'obtenir toutes les autres par des opérations algébriques.

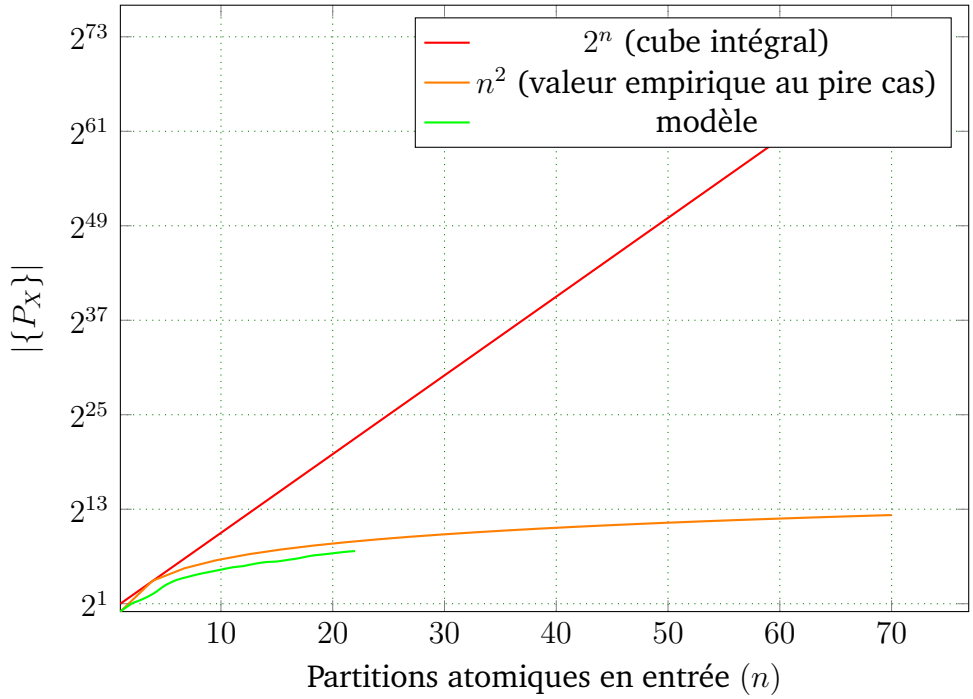


FIGURE 4.11 : Nombre total de partitions annotées à matérialiser (courbe verte) en fonction du nombre n de partitions atomiques en entrée

4.6 Conclusion et perspectives

Dans ce chapitre, nous avons conçu une algèbre de partitions dans le but de gérer des requêtes d'agrégation sur une base de données multidimensionnelle, avec des applications évidentes dans le traitement de requêtes OLAP. Nous avons discuté des propriétés des cartes entre les attributs et leur modèle, sous la forme de partition des tuples. Nous avons également montré qu'il est possible de construire une méthode d'accès sur des vues matérialisées avec un coût de fonctionnement négligeable, à l'aide de notre structure.

Dans un futur proche, une question essentielle sera de mettre en place une politique d'indexation en fonction de la capacité de stockage, c'est-à-dire, en évaluant asymptotiquement le nombre de partitions devant être matérialisées dans le but d'obtenir un compromis entre son coût et celui du traitement des requêtes en direct.

5

Application au consensus de partitions

En classification non-supervisée, le consensus de partitions[1] vise à calculer une unique partition des données qui réalise un compromis parmi les spécificités de chacune des partitions en entrée. La plupart des approches, basées sur des propriétés de graphe ou statistiques, sont reconnues pour leur robustesse et leur capacité à passer à l'échelle.

Dans ce chapitre, nous proposons un point de vue complémentaire des contributions habituelles dans le domaine par l'emploi d'opérations algébriques. Celles-ci permettent d'exposer la nature et les propriétés du raisonnement applicable de manière systématique. Nous fondons notre approche sur les opérations du treillis des partitions, pour lequel nous allons exposer comment certains opérateurs intéressants peuvent être adjoints à la structure dans le but d'exprimer une formule réalisant un compromis parmi toutes les partitions.

Nous allons montrer que l'adoption d'une approche incrémentale peut aider à conserver des agrégats au sein des clusters et qui sont représentatifs parmi les partitions en entrée. Au-delà de la capacité à modéliser des formules, nous relevons également la difficulté à identifier les règles logiques régissant les propriétés des opérations. Cela est dû principalement aux lois de composition des opérateurs du treillis qui influent sur la capacité à tirer des conclusions utiles sur la pertinence du résultat.

[1] *consensus/ensemble clustering* en version originale.

5.1 Introduction

Nous traitons ici de classification non-supervisée ou plus exactement du traitement des partitions qui résultent de l'application d'algorithmes de clustering sur un jeu de données particulier. Nous discutons donc de la manière par laquelle il est possible de combiner les partitions en vue de construire une unique partition qui reflète le mieux l'organisation des données en divers classes.

Notre seule entrée concerne donc un ensemble de partitions définies sur un univers commun ; nous sommes indifférents aux propriétés initiales et aux spécificités des algorithmes qui ont été utilisés pour construire les partitions.

Le point de vue que nous défendons ici, est que chaque partition est donc le modèle associé à une proposition dont l'énoncé nous est inconnu. Par conséquent, puisqu'il est impossible d'exprimer la cohérence des opérations vis-à-vis des propriétés initiales dont chaque partition est le représentant, nous considérons le problème de la recherche d'un compromis parmi toutes les partitions pouvant être engendrées, et dépendant naturellement des propriétés de sa structure de treillis.

Nous allons donc conceptualiser certaines propriétés sur le treillis des partitions dans le but d'en utiliser les opérateurs en lien avec une sémantique particulière, permettant d'en prédire le comportement.

Plusieurs raisons expliquent le choix de réaliser le compromis de manière déterministe :

- La sémantique exprimée par la notion de compromis : une présentation algébrique est susceptible d'exposer plus facilement le sens apporté par chacune des opérations impliquées dans une expression. Par conséquent, cela légitime le fait qu'une instance particulière puisse représenter un ensemble de partitions ;
- La formulation de l'expression réalisant le compromis et son calcul : à l'instar du chapitre précédent, nous souhaitons manipuler des partitions définies sur des univers divers et variés et donc indépendamment de ceux-ci. En dépit du fait que le concept soit omniprésent, il n'existe pas de mise en œuvre pratique permettant d'instancier des partitions au sein d'un environnement de développement et de les combiner. Ce dernier point sera en particulier l'objet du prochain chapitre dans lequel nous allons mettre en œuvre une représentation relationnelle des partitions et les opérateurs afférents (*c.f.* Chapitre 6) ;

Notre principal résultat porte sur l'étude des conditions formelles qui permettent d'effectuer un raisonnement sur les agrégats au sein d'une partition depuis les opérateurs dédiés. Manipuler algébriquement certains objets signifie qu'à tout instant, ce qui est admis comme valide ou cohérent est préservé, quels que soient l'opération impliquée et les opérandes associés.

Nous avons vu qu'en tant que modèle algébrique d'un système logique, les partitions

d'ensemble héritent naturellement des propriétés du calcul associée ; les classes d'équivalence étant alors des relations de congruence, et permettent de simuler les opérations logiques associées (Section 2.4). Comme nous allons le montrer, lorsque les partitions sont elles-mêmes les propositions sur lesquelles on souhaite raisonner, le calcul associé n'est pas congruent et entrave notre capacité à construire un raisonnement valide.

Afin d'être en mesure de conceptualiser finement le raisonnement sur des objets combinatoires, nous avons exploité dans le chapitre précédent les propriétés associées à un théorème de représentation nous permettant d'y plonger et maîtriser les partitions naissant des résultats de requêtes d'agrégation. Dans celui-ci, nous allons étendre l'approche générale énoncée dans la Section (3.3.2), valide uniquement dans le cas des treillis distributifs, mais ne constitue pas une fin en soi.

5.1.1 Les problématiques liées à la classification

Classifier des données selon des sous-populations est en soi une activité fondamentale et prolifique, aussi bien dans les problématiques d'analyse de données pour l'aide à la décision qu'en bio-informatique [70], l'analyse de données audiovisuelles [68], *etc.*

En particulier, cette tâche est le fruit de toujours plus de contributions en raison de l'incapacité d'un seul modèle ou d'une seule approche à s'adapter à tous les jeux de données, voire à s'autoévaluer en raison de la difficulté à discerner une vérité de terrain.

De plus, le nombre de classes adéquat est souvent délicat à déterminer. Les modèles et les techniques proposés dans la littérature se basent généralement sur l'optimisation de critères variés dans le but de réaliser des classes homogènes. La recherche des meilleures bornes suit souvent différentes méthodologies algorithmiques. Dans certains cas, il est possible d'obtenir des modèles différents qui sont la représentation de solutions localement optimales.

Pour ces raisons, la dernière décennie a vu la naissance d'une nouvelle piste de recherche : le clustering d'ensemble ou consensus de partitions. Cette approche cherche à construire des critères et techniques dans le but d'agréger les meilleures facettes associées à de multiples partitions, dans une seule partition : le problème étant que chaque partition est susceptible de modéliser des critères antagonistes.

Il existe deux approches orthogonales lorsqu'on souhaite construire une partition réalisant un compromis :

- la première est fondée sur l'exploitation de connaissances implicites et relatives au champ d'application d'où provient le jeu de données. Ce qui peut alors aider à formuler des hypothèses sur le mécanisme de génération de la partition et puis, sur la vraisemblance des différentes partitions (*c.f.* [79, 135]) ;

- la seconde énonce que la partition optimale est simplement définie comme un compromis entre chaque partition en entrée (*c.f.* [126, 131]).

La modélisation algébrique du problème vise à améliorer la compréhension de la sémantique des mécanismes impliqués lorsque sont combinées les partitions et donc leurs agrégats. En soi, la recherche d'un compromis transcende largement le cas des partitions d'ensemble issues de l'application d'algorithmes de clustering.

La théorie du choix social procure un cadre formel dans lequel la nature du compromis dépend du mélange ou de l'agrégation des diverses opinions ou préférences, représentant un groupe d'individus pour déterminer une décision collective ou commune, celle-ci devant refléter autant que possible les préférences de chaque individu. Le caractère élusif de cette définition favorise l'application de cette approche et de ces principes dans des disciplines dont les champs d'application ne sont pas liés au premier abord (p.ex. [29, 66, 106]).

À quelques exceptions notables [80, 91, 92, 114, 138], le clustering, en dehors du champ algorithmique, ne pourvoit que peu de principes régissant l'agrégation de vecteurs de données. Généralement, on retrouve une modélisation *ad hoc* basée sur les propriétés attendues des données utilisées (images, sons, données géométriques, *etc.*) et qui guide la construction des algorithmes ou la définition des critères à optimiser ; ceux-ci étant généralement subjectifs et contradictoires, car confrontant des points de vue opposés.

5.1.2 La voie axiomatique et ses implications

Le théorème d'impossibilité d'Arrow [8], ou également « paradoxe d'Arrow »est une modélisation mathématique démontrant l'incapacité à mener un processus de choix dont la décision représente, de manière incontestable, l'ensemble des préférences individuelles. La construction d'une telle procédure nécessite alors de contourner la limitation imposée par ces principes en relâchant l'un d'entre eux.

On présume que Ω est l'univers contenant les éléments à agréger, les critères d'agrégations sont généralement définis de la manière suivante.

Indifférence face aux alternatives non pertinentes

Lorsque x, y doivent être agrégés dans la même classe, la décision associée ne doit pas dépendre de la décision prise sur les autres éléments dans $\Omega - \{x, y\}$.

Optimalité suivant Pareto

Si deux éléments x, y sont agrégés ensemble parmi chaque alternative, alors ils le seront également en sortie.

Non-dictature

La décision commune doit tenir compte de toutes les préférences et un choix particulier ne doit pas relever d'un sous-ensemble particulier des préférences.

Généralement, les éléments de Ω sont plongés dans une structure d'ordre qui permet de les décrire et de les comparer formellement sans se soucier de leur formulation ou même de leur évolution lors d'une procédure de décision commune [83, 90, 108].

Par conséquent, il est possible d'exprimer des propriétés sur la nature combinatoire de ces objets à partir des propriétés exprimées sur l'ensemble des objets élémentaires qui les composent.

L'intérêt est de pouvoir définir les propriétés du processus de décision à partir des contraintes combinatoires exposées par la structure qui guide leur composition [96].

Dans [17], Birkhoff et Kiss mettent en évidence les relations suivantes pour tout triplet d'éléments x, y, z dans un treillis L :

$$\begin{cases} \underline{m}(x,y,z) & = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z) \\ \overline{m}(x,y,z) & = (x \vee y) \wedge (x \vee z) \wedge (y \vee z) \end{cases} \tag{5.1}$$

telles que $\underline{m}(x,y,z) \leq \overline{m}(x,y,z)$.

Ces relations bornent alors l'ensemble des éléments qui sont les médianes du triplet. En particulier, ils montrent que, lorsque L est distributif, $m(x,y,z)$ détermine l'unique élément m qui minimise la somme suivante :

$$\sum_{i=1}^{n} d(m,i) \tag{5.2}$$

avec

$$d(x,y) = |\mathcal{J}_{\leq x} - \mathcal{J}_{\leq y}| + |\mathcal{J}_{\leq y} - \mathcal{J}_{\leq x}| \tag{5.3}$$

$$= |\mathcal{J}_{\leq x} \,\Delta\, \mathcal{J}_{\leq y}| \tag{5.4}$$

avec $m = \underline{m} = \overline{m}$ lorsque n est impair[2].

La fonction $d(.,.)$ évalue la distance par le biais de la différence symétrique, soit le nombre de parties élémentaires qui diffèrent entre x et y ; celle-ci pouvant être définie depuis une valuation latticielle isotone et basée sur la fonction de rang.

L'élément médian permet donc de minimiser le nombre d'ajouts ou de retraits d'éléments dans sa décomposition pour atteindre chacun des x_i. Guilbaud [62] fait la relation entre médiane métrique et médiane algébrique à l'aide de la procédure majoritaire – où chaque décision est entérinée par une majorité stricte – lorsque la structure est distributive.

[2]Pour vérifier cette assertion, il suffit d'imaginer que les éléments forment une chaîne au sein de la structure et selon la parité, il y aura soit un seul milieu, soit deux milieux, figurés par les deux bornes.

Barbut généralise ensuite dans [12] cette définition et définit les bornes de l'intervalle des éléments médians de manière constructive. Étant donné un vecteur de préférences $\mathbf{P}$, on a les formules suivantes :

$$\begin{cases} \underline{m}(\mathbf{P}) = \bigvee\limits_{\{X \subset \mathbf{P}, |X| > n/2\}} \bigwedge\limits_{x_i \in X} x_i \\ \overline{m}(\mathbf{P}) = \bigwedge\limits_{\{X \subset \mathbf{P}, |X| > n/2\}} \bigvee\limits_{x_i \in X} x_i \end{cases} \tag{5.5}$$

Il est assez aisé de constater que ces formules définissent un intervalle, au sein duquel sont définis les éléments médians de la structure. De plus, ces fonctions sont des spécialisations des fonctions *unanimité* et *co-unanimité*, définies par les formules suivantes :

$$\begin{cases} \underline{u}(\mathbf{P}) &= \bigwedge\limits_{x_i \in \mathbf{P}} x_i \\ \overline{u}(\mathbf{P}) &= \bigvee\limits_{x_i \in \mathbf{P}} x_i \end{cases} \tag{5.6}$$

La relation suivante est alors vérifiée :

$$\underline{u} \leq \underline{m} \leq \overline{m} \leq \overline{u} \tag{5.7}$$

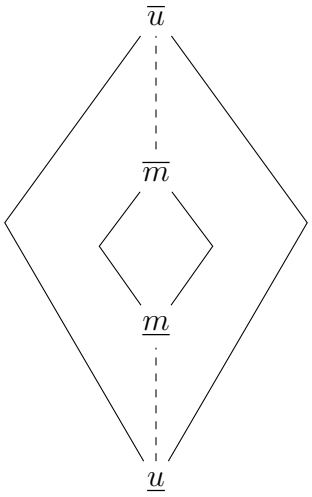


FIGURE 5.1 : Relation d'inclusion entre les intervalles associés aux bornes

L'une des caractéristiques propres à cette approche est donc de dépendre de la décomposition de n'importe quel élément x en un ensemble d'éléments $\{j_i\}$ qui implique x ou, dualement, d'un ensemble d'éléments $\{m_i\}$ subsumés par x.

En particulier, lorsqu'on analyse ces propriétés dans le cadre des treillis complets, tout élément est dès lors présentable à l'aide de plusieurs décompositions. Or, celles-ci ne sont généralement pas uniques.

Lorsque ces structures sont atomistiques ou coatomistiques, donc munies d'une partie génératrice, la nature des décompositions découle alors des relations naturelles qu'entretiennent les générateurs.

Cette situation peut être également visualisée par l'équation de semi-distributivité (*c.f.* p.ex. [98, 113]) suivante :

$$x \vee y_1 = x \vee y_2 = \dots \tag{5.8}$$

$$\implies x \vee \bigwedge_i \{y_i\} = x \vee y_1 \tag{5.9}$$

Cette identité indique que tout élément peut être induit par plusieurs collections de plus petits éléments. Si deux décompositions diffèrent sur des $\{y_i\}$, alors il existe toujours un meilleur élément parmi l'ensemble et donc une borne inférieure. On peut le voir par exemple avec l'ensemble $\{1, 2, 3, 4\}$ qui admet les décompositions $\{1, 2\} \cup \{2, 3, 4\} = \{1, 2\} \cup \{1, 3, 4\}$, l'application de la formule permettant de déduire que $\{1, 2, 3, 4\} = \{1, 2\} \cup \{3, 4\}$.

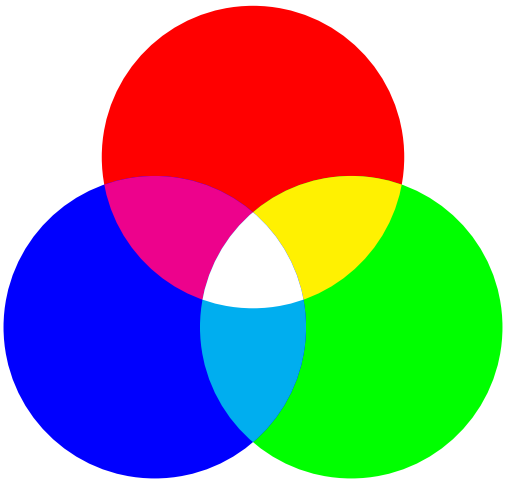


FIGURE 5.2 : Diagramme de Venn des couleurs primaires

Pour illustrer notre propos, nous allons utiliser la figure (5.2). On voit facilement que c'est un diagramme figurant l'ensemble des synthèses entre chaque couleur. Les atomes sont donc ici les couleurs primaires $Rouge, Vert$ et $Bleu$ tandis que la borne supérieure symbolise l'intersection entre deux couleurs. La borne inférieure est elle représentée par la plus grande couleur recouvrante.

Par exemple, $Magenta \wedge Jaune = Rouge$ puisque le rouge est la seule couleur dont elles dépendent (*c.f.* figure 5.3).

Dans cette structure, la définition d'un élément médian détermine donc la couleur la plus proche parmi un ensemble de trois couleurs. Nous aurons par exemple :

$$m(Rouge, Jaune, Vert) = Jaune$$

Ce qui est simple à deviner étant donné que $Jaune$ est obtenue par synthèse additive des deux autres. De plus, on peut calculer que l'élément médian des couleurs primaires est nécessairement le noir tandis que celui des couleurs secondaires est donc le blanc. L'intuition est la suivante : puisque chaque synthèse est réalisée de manière déterministe, les décompositions ne sont pas ambiguës et permettent donc la synthèse soustractive.

Puisque la métrique évalue le résultat à l'aune des différences avec chaque entrée, il pourrait être envisagé de l'exprimer par l'union des différences symétriques :

$$\bigvee_{x_i, x_j \in \mathbf{P}} (x_i \,\Delta\, x_j) \tag{5.10}$$

Ceci n'est bien sûr valable que dans les structures latticielles uniquement complémentées. Or, l'existence d'une complémentation dépend de la distributivité. Ce qui nous amène à notre proposition.

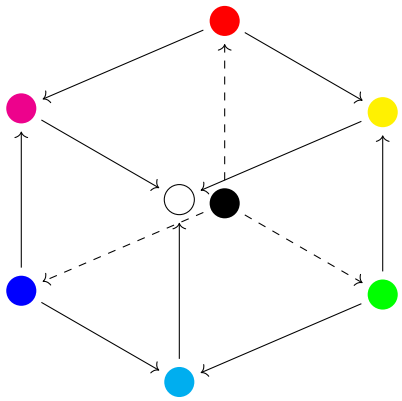


FIGURE 5.3 : Structure des relations entre couleurs

5.1.3 Notre analyse du problème

En l'absence de la propriété de distributivité dans la structure du treillis des partitions, il est délicat d'appliquer la procédure majoritaire sur un ensemble de partitions $\mathbf{P}$, construisant une partition qui possède une forte probabilité d'être très « éloignée » de chaque partition dans $\mathbf{P}$.

En l'occurrence, lorsque la structure est au moins semi-modulaire – propriété qui sera définie peu après – $\underline{m}$ ou $\overline{m}$ représente une borne pour les éléments médians de la structure [82]. Néanmoins, il n'est pas possible à l'heure actuelle, de prédire la dimension de la structure bornée par $\underline{m}$ ou $\overline{m}$ et donc le nombre potentiel d'éléments médians à envisager.

De plus, le problème de la validation d'une partition médiane apparaît ; étant donné que la partition résultante ne reflète pas nécessairement la vérité de terrain. Cela suppose également que la stratégie employée pour engendrer $\mathbf{P}$ peut avoir un impact négatif sur la cohérence de la partition issue de la décision collective, aussi bonne soit-elle ; problématique que nous n'aborderons pas ici.

Notre objectif est de définir des opérations de complémentations dans le treillis des partitions. Or, si cette structure permet la définition d'éléments complémentaires, ceux-ci sont rarement uniques et l'objectif semble être une impasse.

Les équations du type suivant n'admettent donc pas un point fixe :

$$\begin{cases} a \wedge x \leq b \\ a \leq x \vee b \end{cases} \tag{5.11}$$

La première solution que nous avons envisagée dans [43] est la suivante : chaque partition non triviale $P \in \Pi^{\Omega}$ admet plusieurs représentations $\phi(P)$ telles que $|\{\phi(P)\}| > 1$. Puisque nous ne pouvons pas définir de point fixe associé aux équations $P \implies Q$ et $P - Q$, entre deux partitions, nous allons donc restreindre le domaine de définition de chaque opérateur de telle sorte que l'équation associée admette une solution unique.

La seconde solution qui est annoncée sans avoir été développée dans l'article repose sur le questionnement suivant : comment réaliser des opérateurs en s'affranchissant de l'absence de la propriété de distributivité ?

Dans le chapitre précédent, les partitions annotées sont considérées comme les modèles de requêtes projectives sur une relation dans une base de données telles que les opérations sur les partitions reflètent exactement les opérations permises sur les relations.

Dans le cas où les partitions sont des résultats de clustering, d'une part nulle annotation ne permet de différencier une méthodologie plutôt qu'une autre ayant mené à ce type d'agrégation, d'autre part, l'absence d'axiomatisation pour le clustering rend plus laxiste la combinatoire des partitions, car ces dernières ne matérialisent pas des relations de congruence sur un univers donné (*c.f.* Section 2.4).

En particulier, le théorème de représentation (*c.f.* Section 3.3.2) ne s'applique plus dans ce cas. La dualité suivante :

$$a \vee b \triangleq \neg(\neg a \wedge \neg b) \tag{5.12}$$

qui est obtenue par les méthodologies initiées par Priestley et Stone [110, 125] n'est pas satisfaite. Nous allons donc appliquer la dualité présentée par Urquhart [133] pour définir des opérations de complémentation entre partitions d'ensemble.

Nous mettrons en perspective ces résultats dans le cadre de la représentation des treillis abordée à la fin du Chapitre 2 dont nous nous servirons pour formuler notre proposition.

5.2 Sur le treillis des partitions

Dans cette section, nous allons introduire les principales propriétés du treillis des partitions. Dans ce but, nous allons présenter la relation qu'entretiennent entre eux les compléments. À partir de celles-ci, nous tenterons de définir des opérations de complémentation compatibles avec la structure, et que nous illustrerons en proposant une procédure pour calculer une partition médiane.

5.2.1 Représentation des partitions

La relation, qui lie les générateurs d'un treillis L et les différentes représentations qu'ils induisent, est largement connue [40]. Celle-ci permet de caractériser, en premier lieu, la dimension d'une représentation de n'importe quel élément $x \in L$ en fonction de ses générateurs. La représentation $\phi(x)$ du chapitre 3 lorsqu'elle est restreinte à la projection sur les $\vee$-irréductibles, construit donc un sous-ensemble de générateurs $A \subseteq \mathcal{J}(L)$ tel que :

$$x = \bigvee \{j \in A\} \tag{5.13}$$

Celle-ci est redondante s'il existe un sous-ensemble $A' \subset A$ tel que $x = \bigvee \{j \in A'\}$.

Inversement, celle-ci est minimale si $x < \bigvee \{j \in A'\}$ et lorsque le treillis est atomistique – les $\vee$-irréductibles sont tous des atomes – $\bigvee \{j \in A'\} = \bot$. Dans le treillis des partitions, une définition explicite de cette représentation est réalisable de la manière suivante :

$$\phi : \Pi^{\Omega} \to \mathcal{P}(\Omega^2) \tag{5.14}$$

$$P \mapsto \sim_P \triangleq \bigcup_{a \in C \in P} a \times a \tag{5.15}$$

Celle-ci construit la relation d'équivalence associée à chaque classe de $\phi(P)$. Chaque élément dans la relation est donc, à l'exception des éléments triviaux[3] (réflexifs et symétriques) précisément un atome de $\mathcal{J}(\Pi^{\Omega})$.

Posons, par exemple, la partition $P = 124|3$ dans le treillis des partitions à quatre éléments (*c.f.* figure 5.4), on voit nettement qu'elle est définissable à l'aide des atomes $12|3|4$, $14|2|3$ et $1|24|3$, ce qui nous donne :

$$P = 12|3|4 \vee 14|2|3 \vee 1|24|3 \iff \sim_P = \{12, 14, 24\} \tag{5.16}$$

Or, on remarque aisément qu'il existe des parties de $A \subseteq \sim_P$ telles que $\phi^{-1}(A) = P$, ce qui nous donne la relation suivante entre la représentation et ces différentes solutions :

$$\phi \circ \phi^{-1} : \mathcal{P}(\Omega^2) \to \mathcal{P}(\mathcal{P}(\Omega^2)) \tag{5.17}$$

$$\sim_P \mapsto \{X \in \mathcal{P}(\Omega^2) \mid \phi^{-1}(X) = \phi^{-1}(\sim_P)\} \tag{5.18}$$

Concernant la partition P, on obtient donc :

$$\phi \circ \phi^{-1}(\sim_P) = \{\{12, 14\}, \{12, 24\}, \{14, 24\}, \{12, 14, 24\}\} \tag{5.19}$$

La structure résultante est donc un demi-treillis pour l'inclusion dont $\sim_P$ est la borne supérieure. Elle représente également les classes d'équivalence de la partition P dans $\mathcal{P}(\Omega^2)$; l'ensemble de tous les modèles équivalent par la fermeture transitive.

[3] Par la suite, ces éléments ne seront pas explicités lorsque nous introduirons la relation d'équivalence, associée à une partition.

Il s'agit maintenant de savoir si ces représentations sont congruentes, c.-à-d. si le choix de n'importe laquelle préserve la condition suivante (*c.f.* également la définition 2.4.4).

Définition 5.2.1 (Convexité)**.** *Soit une partition $P \in \Pi^{\Omega}$, on définit la relation d'équivalence $\equiv$ sur $\phi \circ \phi^{-1}(\sim_P)$. $\equiv$ est une relation de congruence si et seulement si pour tout Q :*

$$X \equiv \sim_P \implies \begin{cases} X \cap \sim_Q = \sim_P \cap \sim_Q \\ X \cup \sim_Q = \sim_P \cup \sim_Q \end{cases} \tag{5.20}$$

Les ensembles minimaux sont donc tous des générateurs pour P, néanmoins, l'absence de critère algébrique pour choisir l'un d'entre eux, ne permet pas de définir une représentation minimale pour P.

Posons par exemple, $Q = 1|24|3$ telle que $\sim_Q = \{24\}$. Le résultat de $P \wedge Q$ ou $\sim_P \cap \sim_Q$ dépend de la représentation de P :

$$\begin{aligned} \{12, 14\} \cap \{24\} &= \varnothing \\ \{12, 24\} \cap \{24\} &= \{24\} \\ \{14, 24\} \cap \{24\} &= \{24\} \\ \{12, 14, 24\} \cap \{24\} &= \{24\} \end{aligned}$$

Il apparaît que seule la représentation maximale issue de $\phi(P)$ est stable pour l'intersection. En particulier, la loi semi-distributive n'est donc pas respectée et on a :

$$P = \bigvee_{a \in \{12,24\}} a = \bigvee_{b \in \{14,24\}} b \tag{5.21}$$

Or,

$$\bigvee_{(a,b) \in \{12,24\} \times \{14,24\}} (a \wedge b) = \{24\} \tag{5.22}$$

n'est pas une représentation valide de P. Cependant, il apparaît que la représentation maximale s'obtient depuis toutes les autres à travers la fermeture transitive $\langle . \rangle^+$ puisque chacune d'entre elles conserve l'accessibilité de n'importe quel élément vers un autre appartenant à la classe.

On définit maintenant $P = 123|4$ et $Q = 1|2|34$, étant donnée leur représentation maximale respective sous la forme de relation d'équivalence, on a alors :

$$\{12, 13, 23\} \cup \{34\} = \{12, 13, 23, 34\}$$

L'ensemble résultant ne reflète pas l'ensemble des associations résultant de l'application de $P \vee Q$. Simuler le calcul de la borne supérieure nécessite d'appliquer après l'union de deux relations d'équivalence, la fermeture transitive de celle-ci, telle que :

$$\sim_{P \vee Q} = \{12, 13, 14, 23, 24, 34\}$$

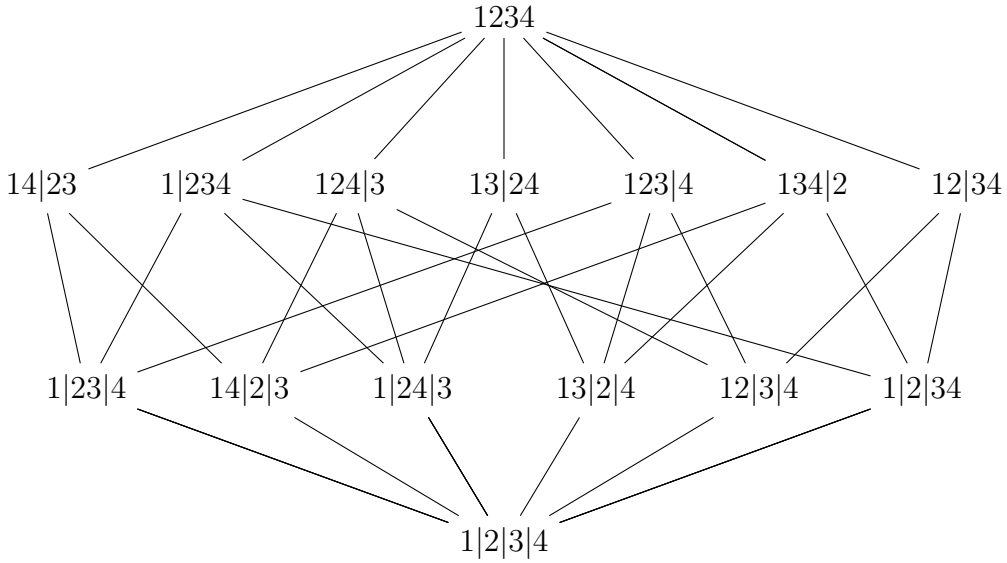


FIGURE 5.4 : Le treillis des partitions $\Pi^{\{1,2,3,4\}}$

Lorsqu'il existe une représentation minimale des éléments dans une structure algébrique munie de l'inclusion, cela permet donc de représenter ceux-ci comme un ensemble invariant d'atomes. Dans le cas contraire, cela spécifie que chaque élément peut être construit de différentes manières.

Logiquement parlant, chaque partition est le modèle figurant une séquence d'agrégations des éléments de l'univers. Chaque représentation, autre que maximale, procure donc une interprétation légèrement différente de celle-ci, vis-à-vis des autres partitions construites.

Une partition, qui n'est pas réduite à une collection d'atomes disjoints, ne permet donc pas, en l'état, de mesurer congrument la séquence exacte des atomes qui ont été agrégés ensemble. Posons par exemple, $P = 1234|567$, la première classe de P admet donc les deux scénarii suivants, $\langle 13, 14, 24\rangle$ et $\langle 23, 12, 34\rangle$, qui sont tous les deux plausibles sans être pourtant corrélés.

On a donc le théorème suivant.

Théorème 5.2.1. *[104] Le treillis des partitions n'est pas distributif.*

Cela souligne donc l'interdépendance des éléments irréductibles dans la définition d'une partition d'ensemble, nécessitant de fait de choisir la plus grande relation recouvrant ceux-ci.

Définition 5.2.2 ([38])**.** *Soit un treillis quelconque L et deux $\vee$-irréductibles $j, j' \in \mathcal{J}(L)$, on définit la relation C sur $\mathcal{J}(L)$ telle que :*

$$(j, j') \in C \iff \exists x \in L, j \leq x \vee j' \text{ et } j \not\leq x \vee j'' \tag{5.23}$$

telle que $j'' < j'$.

Lorsque le treillis est atomistique, la relation se simplifie suivant l'équation suivante :

$$(j, j') \in C \iff \exists x \in L, j \leq x \vee j' \text{ et } j \not\leq x \tag{5.24}$$

Dans un treillis distributif, lorsque l'élément x désigne un $\wedge$-irréductible, cette définition permet d'énoncer un analogue du lemme de séparation à l'aide de ces relations.

Cette relation est également semblable à la relation $\delta \subseteq \mathcal{J}(L) \times \mathcal{J}(L)$ définie dans [95] lorsque le treillis est atomistique. Par exemple, la structure d'ordre pour la relation de divisibilité construit une relation entre les atomes et les $\vee$-irréductibles :

$$C = \{(2,4), (2,8), \ldots, (3,6), (3,9), \ldots\}$$

Définition 5.2.3 (Perspectives)**.** *Soit un treillis L, étant donné un élément $\vee$-irréductible j et un élément $\wedge$-irréductible m, on définit les relations sur $\mathcal{J}(L) \times \mathcal{M}(L)$.*

$$\not\nearrow j = \{x \mid j \not\leq x\} \tag{5.25}$$
$$\not\swarrow m = \{x \mid x \not\leq m\} \tag{5.26}$$

Contrairement à l'usage [136], nous utilisons les symboles barrés pour dissocier ces éléments de la définition des filtres et idéaux principaux. Un choix qui sera justifié dans les prochaines sections. On note également que ces définitions sont généralisables pour des ensembles, à l'instar des cônes (*c.f.* définition 3.1.21).

On constate aisément à partir de ces relations, qu'étant données $j \in \mathcal{J}(L)$ et $m \in \mathcal{M}(L)$, on a les propriétés suivantes :

$$j \wedge x = \bot, \ \forall x \in \not\nearrow j \tag{5.27}$$
$$m \vee x = \top, \ \forall x \in \not\swarrow m \tag{5.28}$$

La perspective permet donc de mettre en relation les éléments qui sont mutuellement orthogonaux [67, 75].

Définition 5.2.4 (Complémentation)**.** *Soit un treillis borné $(L, \leq, \wedge, \vee)$, pour tout $a \in L$, on définit le complément de a dans L, les éléments a^* et a_* qui vérifient les conditions suivantes :*

$$\begin{cases} a \wedge a_* = \bot \\ a \vee a^* = \top \end{cases} \tag{5.29}$$

La condition de séparabilité, qui s'applique dans les treillis distributifs, implique l'unicité de la relation entre a et ses compléments. Par conséquent, la dualité entre les ensembles générateurs $\mathcal{J}(L)$ et $\mathcal{M}(L)$ permet d'affirmer le théorème suivant.

Théorème 5.2.2. *Soit un treillis distributif L, pour tout élément $x \in L$, le couple $(x_*, x^*) \in \mathcal{J}(L) \times \mathcal{M}(L)$ est unique. En particulier, x^* est maximal dans $\not\nearrow x$ et x_* est minimal dans $\not\swarrow x$.*

Dans le cas contraire, le filtre engendré par un élément irréductible, $\uparrow j$, n'induit pas une bipartition de L telle que pour un complément donné m :

$$\uparrow j \cap \downarrow m = \varnothing \tag{5.30}$$

$$\uparrow j \uplus \downarrow m \subseteq L \tag{5.31}$$

Présumons, par exemple, que la partition $P = 12|3|4$ est un modèle, alors l'homomorphisme $h : \Pi^{\Omega} \to \{0, 1\}$, qui est défini par :

$$h(x) = \begin{cases} 0 & \text{si } x \in \downarrow P^* \\ 1 & \text{si } x \in \uparrow P \end{cases} \tag{5.32}$$

est nécessairement incomplet.

En effet, en l'absence de la propriété de distributivité, pour tout élément x dans un treillis $(L, \leq)$, les éléments maximaux, respectivement minimaux de l'ensemble $\not\uparrow x$, respectivement dans $\not\downarrow x$, ne sont pas ordonnables suivant la relation d'ordre de la structure et ne peuvent être départagés pour établir un point fixe.

Cela implique donc que seul l'ensemble des idéaux engendrés par chaque P^* permet de couvrir Π^{Ω} tel que :

$$\uparrow P \uplus \bigcup_{P^* \in \max \not\uparrow P} \downarrow P^* = \Pi^{\Omega} \tag{5.33}$$

Puisque $\not\uparrow P = \bigcup_{P^* \in \max \not\uparrow P} \downarrow P^*$, on a le corollaire suivant :

Corollaire 5.2.1. *Un treillis L est distributif si et seulement si pour tout élément $x \in L$, x^* et x_* sont uniques et :*

$$\begin{cases} \not\uparrow x & = \downarrow x^* \\ \not\downarrow x & = \uparrow x_* \end{cases} \tag{5.34}$$

et qui permet d'énoncer l'identité due à De Morgan :

$$\begin{cases} (a \wedge b)^* = a^* \vee b^* \\ (a \vee b)_* = a_* \wedge b_* \end{cases} \tag{5.35}$$

Dans le cas du treillis des partitions, on a donc le résultat suivant.

Corollaire 5.2.2. *Pour toute partition P non triviale dans le treillis des partitions, P a plus d'un complément.*

Il n'est donc pas possible *a priori* de construire le complément relatif d'une partition en l'absence d'un critère algébrique adéquat permettant de restreindre l'espace des solutions.

Nous allons donc dans un premier temps modifier la sémantique de la complémentation afin d'arriver à nos fins.

5.2.2 Filtrage des partitions

Comme nous venons de le voir, la dualité induite par l'identité de De Morgan entre les expressions conjonctives et disjonctives empêche la sélection d'un complément unique. En particulier, pour chaque complément, il est possible d'engendrer l'idéal correspondant afin de couvrir l'ensemble Π^Ω.

Puisqu'il nous semble impossible de construire une seule solution pour l'équation de la semidistributivité, définissant formellement chaque complément, nous choisissons de restreindre le domaine des solutions admissibles à une portion contenant un point fixe.

Définition 5.2.5 (Opérateur de différence)**.** *Soit un treillis L, l'opérateur de différence est défini par la formule suivante :*

$$a - b = \bigwedge \{x \mid a \leq x \vee b\} \tag{5.36}$$

dont le résultat est le plus petit élément devant être adjoint à b tel que :

$$a \vee (a - b) = b \tag{5.37}$$

Ceci nécessite donc de calculer le résultat de :

$$a \vee \bigwedge_{s \in S} s \tag{5.38}$$

où S est l'ensemble solution $\{x \mid a \leq x \vee b\}$. Dans Π^Ω, il nous est impossible de calculer $P - Q$ de la sorte. Nous avons retenu comme solution de restreindre l'ensemble S à $\mathcal{P}(P)$. En effet, il nous est possible d'établir une correspondance entre les classes d'une partition et les atomes d'une algèbre booléenne.

Par exemple, la figure 5.5 présente le treillis engendré par la partition $P = 123|456|789$. C'est un cas particulier de l'exemple donné dans le chapitre 3 (*c.f.* la figure 3.14(a)) qui montre la complétion du treillis engendré par une collection $X \in \mathcal{P}(\mathcal{P}(\Omega))$.

Utiliser ce stratagème revient à considérer que les partitions sont les modèles d'un calcul booléen des propositions dont les classes sont les atomes ; celles-ci engendrant toutes les combinaisons possibles.

À cet égard, les opérations latticiels entre les partitions acquièrent donc les mêmes propriétés que les modèles qu'elles sont censées représenter ; cas similaire au chapitre précédent où l'ensemble de toutes les partitions annotées $\Pi^\Omega \times \mathcal{P}(\mathcal{A})$ est isomorphe à $\mathcal{P}(\mathcal{A})$.

Suivant cette représentation, on peut alors définir par analogie avec l'algèbre des parties d'un ensemble, des opérateurs adjoints. La différence admet généralement une représentation sous la forme suivante :

$$X - Y = X \cap Y^c \tag{5.39}$$

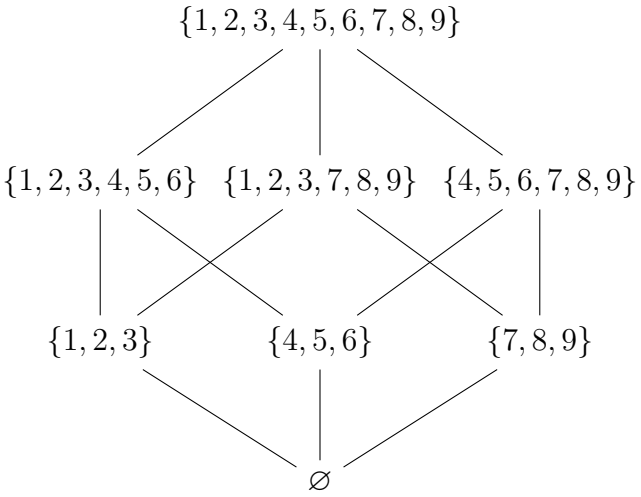


FIGURE 5.5 : Le treillis des parties $\mathcal{P}(P)$ engendré par les classes de $P = 123|456|789$

où $X, Y \in \mathcal{P}(\Omega)$. Le résultat s'obtient donc naturellement par le retrait de tous les éléments qui sont dans X et non dans Y. De plus, il est connu que la définition de l'implication est, dans ce contexte, duale de celle de la différence telle que :

$$(X - Y)^c = X \implies Y \tag{5.40}$$

$$\iff X - Y = Y \implies X \tag{5.41}$$

et seul l'un des deux opérateurs revêt un intérêt.

Étant donné que la complémentation n'existe pas dans Π^Ω et *a fortiori* dans $\mathcal{P}(\Omega^2)$, nous allons nous concentrer sur la définition de la différence.

En effet, l'application de la complémentation ensembliste dans $\mathcal{P}(\Omega^2)$ ne préserve pas leur structure et nécessite donc l'application d'une fermeture algébrique, ce qui est plus naturel que d'invoquer une opération d'ouverture (fermeture duale) dans le cas de l'implication. Cette dernière sera définie en inversant l'apparition des opérandes comme ci-dessus.

L'opérateur est défini de la manière suivante dans $\mathcal{P}(\Omega^2)$:

$$[.]_P - [.]_Q \triangleq \langle [.]_P \cap [.]_Q^c \rangle^+ \tag{5.42}$$

Posons, par exemple, les partitions $P = 1234|567$ et $Q = 123|4567$ et calculons $P - Q$. Les classes de chaque partition peuvent être aisément représentées sous la forme de matrices à coefficients dans $\{0, 1\}$ telles que :

$$\begin{pmatrix} 1&1&1&1&0&0&0\\ 1&1&1&1&0&0&0\\ 1&1&1&1&0&0&0\\ 1&1&1&1&0&0&0\\ 0&0&0&0&1&1&1\\ 0&0&0&0&1&1&1\\ 0&0&0&0&1&1&1 \end{pmatrix} - \begin{pmatrix} 1&1&1&0&0&0&0\\ 1&1&1&0&0&0&0\\ 1&1&1&0&0&0&0\\ 0&0&0&1&1&1&1\\ 0&0&0&1&1&1&1\\ 0&0&0&1&1&1&1\\ 0&0&0&1&1&1&1 \end{pmatrix} \tag{5.43}$$

Le calcul se déroule comme suit :

$$P - Q = \left\langle \begin{pmatrix} 1&1&1&1&0&0&0\\ 1&1&1&1&0&0&0\\ 1&1&1&1&0&0&0\\ 1&1&1&1&0&0&0\\ 0&0&0&0&1&1&1\\ 0&0&0&0&1&1&1\\ 0&0&0&0&1&1&1 \end{pmatrix} \cap \begin{pmatrix} 1&1&1&0&0&0&0\\ 1&1&1&0&0&0&0\\ 1&1&1&0&0&0&0\\ 0&0&0&1&1&1&1\\ 0&0&0&1&1&1&1\\ 0&0&0&1&1&1&1\\ 0&0&0&1&1&1&1 \end{pmatrix}^{c\ +} \right\rangle \tag{5.44}$$

$$= \left\langle \begin{pmatrix} 1&1&1&1&0&0&0\\ 1&1&1&1&0&0&0\\ 1&1&1&1&0&0&0\\ 1&1&1&1&0&0&0\\ 0&0&0&0&1&1&1\\ 0&0&0&0&1&1&1\\ 0&0&0&0&1&1&1 \end{pmatrix} \cap \begin{pmatrix} 0&0&0&1&1&1&1\\ 0&0&0&1&1&1&1\\ 0&0&0&1&1&1&1\\ 1&1&1&0&0&0&0\\ 1&1&1&0&0&0&0\\ 1&1&1&0&0&0&0\\ 1&1&1&0&0&0&0 \end{pmatrix}^{+} \right\rangle \tag{5.45}$$

$$= \left\langle \begin{pmatrix} 0&0&0&1&0&0&0\\ 0&0&0&1&0&0&0\\ 0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0 \end{pmatrix}^{+} \right\rangle \tag{5.46}$$

$$= \begin{pmatrix} 1&1&1&1&0&0&0\\ 1&1&1&1&0&0&0\\ 1&1&1&1&0&0&0\\ 1&1&1&1&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0 \end{pmatrix} = 1234|5|6|7 \tag{5.47}$$

On voit assez nettement que subsistent dans P les classes qui ne sont pas totalement recouvertes par les classes de Q.

Lemme 5.2.1. *Soit deux partitions P et Q, pour tout $C \in P$ et $D \in Q$, on a l'identité suivante :*

$$C \supsetneq D \iff \langle [.]_C \cap [.]_D^c \rangle^+ = [.]_C \tag{5.48}$$

Démonstration. l'identité de droite indique que $[.]_C \cap [.]_D^c$ préserve la fermeture transitive de $[.]_C$, soit l'accessibilité de tous les éléments de l'univers, restreint à la classe C. Si $C = \{1,2,3,4\}$, alors la plus petite relation X dans $\mathcal{P}(\Omega^2)$ est $\{(1,2),(2,3),(3,4)\}$ – à une énumération près des identifiants des éléments – telle que :

$$\langle \{(1,2),(2,3),(3,4)\} \rangle^+ = \{1,2,3,4\}^2 \tag{5.49}$$

La classe associée n'est pas dans le résultat de $P - Q$ si et seulement si :

$$\{(1,2),(2,3),(3,4)\} \cap [.]_D^c = \varnothing \tag{5.50}$$
$$\iff \{(1,2),(2,3),(3,4)\} \not\subset [.]_D^c \tag{5.51}$$
$$\iff \{1,2,3,4\} \subseteq D \tag{5.52}$$

Donc toute classe D strictement incluse dans C préserve la fermeture transitive de celle-ci. □

Corollaire 5.2.3. *Soit deux partitions P, Q, on a l'identité suivante :*

$$P \vee Q = (P - Q) \vee (Q - P) = P \oplus Q \tag{5.53}$$

où $\oplus$ désigne l'opérateur de somme disjointe ou la différence symétrique. C'est une conséquence directe de la définition de l'adjonction : les classes maximales de chaque partition sont épargnées et préservent donc l'accessibilité de n'importe quel élément dans la fermeture transitive.

L'opérateur admet donc une définition plus classique :

$$P - Q \triangleq \{C \in \mathcal{P}(\Omega) \mid C \in P, \forall C' \in Q, C \supsetneq C'\} \tag{5.54}$$

et on a : $Q \to P = P - Q$. Étant donnée leur équivalence, nous allons nous restreindre à la définition de l'implication[4] par la suite.

Par définition, on a les identités suivantes sur les bornes :

$$\top \to \bot = \bot \tag{5.55}$$
$$\bot \to \top = \top \tag{5.56}$$

Considérant la relation de raffinement, on a donc :

$$P < Q \iff \begin{cases} P \to Q = Q \\ Q \to P = \bot \end{cases} \tag{5.57}$$

En particulier, $P \to P = \bot$

Étant donnée une collection de partitions $\mathbf{P}$, si celle-ci forme une chaîne dans Π^Ω, on aura alors :

$$\bigoplus_{P_i, P_j \in \mathbf{P}} = \sup \mathbf{P} \tag{5.58}$$

Par conséquent, on a :

$$\bigoplus \bigwedge_{\mathbf{Q} \subset \mathbf{P}} \mathbf{Q} = \bigvee \bigwedge_{\mathbf{Q} \subset \mathbf{P}} \mathbf{P} \tag{5.59}$$

[4] Le symbole $\to$ désigne l'opérateur algébrique tandis que $\implies$ désigne la méta-implication.

Théorème 5.2.3. *Soit les partitions P, Q, R, l'identité suivante est vérifiée :*

$$P \leq Q \iff \begin{cases} R \to P \leq R \to Q \\ P \to R \geq Q \to R \end{cases} \tag{5.60}$$

Démonstration. Etant donné que $P \leq Q$, chaque classe dans $P \vee Q$ couvre alors celles de P. Il n'existe donc pas de classe $C' \in Q$ telle qu'il existe $D \in R$ et $C \in P$ telles que $C' \subseteq D$ et $C \supset D$ puisque $C \subset C'$. La seconde identité se démontre de la même manière. □

L'implication/la différence est donc compatible avec la relation de raffinement entre deux partitions.

Théorème 5.2.4. *Soient les partitions P, Q, R, l'identité suivante est vérifiée :*

$$R \to (Q \to P) = (R \to P) \wedge (Q \to P) \tag{5.61}$$

Nous allons maintenant utiliser notre opérateur pour réaliser un filtrage sur une collection de partitions $\mathbf{P}$ puis construire une fonction de consensus sur celles-ci. On rappelle en premier lieu la définition des fonctions $\underline{u}(.)$ et $\underline{m}(.)$. La fonction unanimité nécessite de calculer la borne commune à l'ensemble des partitions. Elle est définie de la manière suivante :

$$\underline{u} = \bigwedge_i P_i \tag{5.62}$$

Si celle-ci met en œuvre l'ensemble des propriétés d'Arrow, il est peu vraisemblable qu'elle contienne suffisamment d'agrégats pour permettre une exploitation de son résultat, car beaucoup trop stricte. La fonction suivante réalise un compromis en opérant l'union des consensus obtenues pour chaque majorité formée sur $\mathbf{P}$ [14] :

$$\underline{m} = \bigvee \bigwedge_{\mathbf{Q} \subset \mathbf{P}, |Q| \geq \frac{|P|}{2}} \mathbf{Q} \tag{5.63}$$

On donne également la règle duale :

$$\underline{m} = \bigwedge \bigvee_{\mathbf{Q} \subset \mathbf{P}, |Q| \geq \frac{|P|}{2}} \mathbf{Q} \tag{5.64}$$

Notre proposition consiste, en premier lieu, à calculer un ensemble $\mathbf{P}'$ tel que chaque partition en son sein, soit la version filtrée de celle dans $\mathbf{P}$. Puis, il s'agit d'appliquer l'implication sur chaque partition, avec toutes les autres en prémisses :

$$\bigwedge_{\forall i \neq j} (P_i \to P_j) \tag{5.65}$$

Celle-ci conserve globalement l'ensemble des agrégats maximaux, et supprime toutes les classes qui sont incluses dans au moins une autre classe.

Dans ce cas, si l'on considère $\mathbf{P}$ comme une famille d'ensembles dans $\mathcal{P}(\mathcal{P}(\Omega))$ dont les membres sont les classes de chaque partition, $\mathbf{P}'$ est alors une famille de Sperner où la connectivité entre les classes dépend de leurs intersections mutuelles. En particulier, chaque partition est donc un modèle minimal préservant la couverture de l'ensemble en conservant chaque agrégat apparaissant au moins une fois dans chaque partition.

Dans le but de préserver les agrégats dans le résultat, nous optons pour le maintien des bornes inférieures binaires entre chaque partition dans $\mathbf{P}'$ représentant alors les points dominants dans la structure d'hypergraphe correspondante. On note $transv(.)$ la fonction suivante :

$$transv(\mathbf{P}) \triangleq \bigvee_{(P,Q)\in\mathbf{P}'\times\mathbf{P}'} (P \wedge Q) \tag{5.66}$$

avec :

$$\mathbf{P}' = \{\bigwedge_{i\neq j}(P_i \implies P_j) \mid \forall j \in [1, |\mathbf{P}|]\} \tag{5.67}$$

Afin d'illustrer notre fonction, on définit la collection $\mathbf{P}$ de la manière suivante :

$$\begin{aligned} P_1 &= 147|2|356 \\ P_2 &= 1234|57|6 \\ P_3 &= 126|3|47|5 \\ P_4 &= 123|4567 \\ P_5 &= 124|35|67 \end{aligned}$$

puis sur le multi-ensemble $\mathbf{P}_m = \mathbf{P} \cup \{P_1\}$.

La table (5.1) résume les résultats obtenus pour chaque fonction sur l'ensemble $\mathbf{P}$.

Collection	$\underline{u}$	$\underline{m}$	$\overline{m}$	$transv$
$\mathbf{P}$	$\bot$	$124\|3\|5\|6\|7$	$\top$	$1247\|56\|3$
$\mathbf{P}_m$	$\bot$	$1247\|3\|5\|6$	$\top$	$1247\|56\|3$

TABLE 5.1 : Partition retournée pour chaque méthode

Au premier abord, notre approche semble accomplir une meilleure estimation que la procédure majoritaire. Étant donné que $P - P = P \to P = \bot$ notre procédure renvoie donc un résultat identique lorsque la première partition est dupliquée.

De plus, si l'on compare l'ensemble des paires de Ω^2 maintenues dans chaque partition, on obtient un score global satisfaisant comme l'illustre la table (5.2). En moyenne, chaque paire d'éléments qui sont classés ensemble, apparaît dans la moitié des partitions de l'ensemble, ce qui figure un bon résultat. Néanmoins, on constate que la fermeture transitive, réalisée à la fin du traitement lors du calcul de la borne supérieure, induit le classement d'éléments qui ne devraient pas l'être, à l'instar de $(1, 7)$ ou $(2, 7)$.

	$\underline{m}$		$transv$	
	$\mathbf{P}$	$\mathbf{P}_m$	$\mathbf{P}$	$\mathbf{P}_m$
médiane	$\frac{4}{5}$	$\frac{4}{6}$	$\frac{2}{5}$	$\frac{2}{6}$
moyenne	$\frac{2}{3}$	$\propto \frac{2}{3}$	$\frac{1}{2}$	$\propto \frac{1}{2}$

TABLE 5.2 : Pour chaque paire $(x, y) \in \Omega^2$, moyenne et médiane de la fréquence d'apparitions dans les partitions de chaque collection

Au prix de certains efforts, nous avons montré dans cette section qu'il était possible de construire de nouveaux opérateurs sur le treillis des partitions, en dépit de l'absence de la propriété de distributivité.

Dans la prochaine section, nous allons proposer une méthode de représentation explicite des partitions en fonction de leur complément. Nous allons donc étendre la relation d'ordre naturelle afin de classer simultanément les éléments et leurs complémentaires. Ceci nous permettra de caractériser des propriétés du treillis des partitions en fonction de ces éléments.

5.3 Dualité sur le treillis des partitions

Dans la section précédente, nous avons vu que les éléments $\wedge\vee$-irréductibles présentaient, d'un point de vue logique, des propriétés non triviales. En particulier, les propriétés de la relation $\mathcal{J}(L) \times \mathcal{M}(L)$ entre ces éléments régissent donc certaines propriétés du treillis L.

De plus, ils permettent l'extraction de parties de L préservant les propriétés de la structure ; mettant en évidence, des relations de congruence [88]. Un outil élémentaire pour mettre en avant les propriétés des irréductibles est la structure d'ordre induite sur cette relation [87] et que nous allons étudier.

5.3.1 Le treillis des antichaînes

Définition 5.3.1 (Antichaîne)**.** *Soit un treillis quelconque* $(L, \leq)$*, une* antichaîne $\overline{C} \in \mathcal{P}(L)$ *est un ensemble vérifiant les conditions suivantes :*

$$x \in \overline{C} \iff \forall y \in \overline{C}, x \not\leq y \not\leq x \tag{5.68}$$

Ce qui se simplifie aisément en :

$$x \in \overline{C} \iff \forall y \in \overline{C}, x \leq y \implies x = y \tag{5.69}$$

On notera $\overline{C}(L)$, l'ensemble de toutes les antichaînes d'un treillis L, l'ensemble $\overline{C}(L)$ est donc une collection d'antichaînes et un élément de $\mathcal{P}(\mathcal{P}(L))$.

Par exemple, dans le treillis des partitions à quatre éléments, l'ensemble

$$\{14|23, 1|24|3, 12|34\}$$

est donc une antichaîne. On remarque que les éléments ne sont pas nécessairement tous situés au même niveau.

Lemme 5.3.1. *Soit un treillis L, $\mathcal{J}(L)$ et $\mathcal{M}(L)$ sont donc des antichaînes.*

Il est dès lors facile de voir que toutes les combinaisons d'éléments dans chaque ensemble engendrent alors une antichaîne.

On va montrer par la suite que $\overline{C}(L)$ possède une structure héritée de L.

Définition 5.3.2. *Étant données deux antichaînes $X, Y \in \overline{C}(L)$, l'identité suivante définit une relation d'inclusion :*

$$X \preceq Y \iff \forall x \in X, \exists y \in Y, x \leq y \tag{5.70}$$

Cette définition est assez semblable à celle de la relation de raffinement entre deux partitions où les classes sont disjointes, mais ordonnées pour l'inclusion. Il est donc aisé de vérifier la propriété suivante.

Lemme 5.3.2. *Étant données deux antichaînes $X, Y \in \overline{C}(L)$, on a :*

$$X \subseteq Y \implies X \preceq Y \tag{5.71}$$

Il suffit d'observer qu'une antichaîne induit au minimum la relation réflexive $\forall x \in X, x \leq x$. Cependant, il est important de noter que si les éléments de deux antichaînes sont ordonnés, cela ne signifie pas que $X \subseteq Y$.

On introduit maintenant les propriétés prouvant que $(\overline{C}(L), \preceq)$ est une relation d'ordre partiel.

Théorème 5.3.1. *Étant données les antichaînes $X, Y, Z \in \overline{C}(L)$, on a les propriétés suivantes :*

1. $X \preceq X$ *(réflexivité)*
2. $X \preceq Y$ *et* $Y \preceq X \implies X = Y$ *(antisymétrie)*

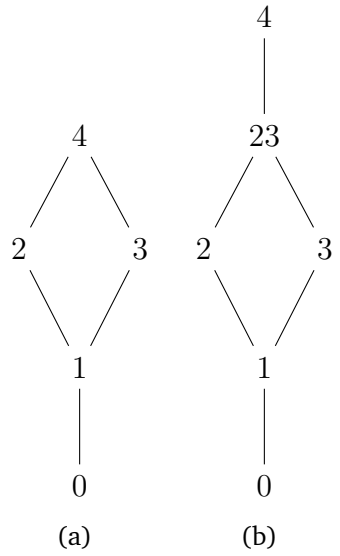


FIGURE 5.6 : (a) Un ordre partiel L avec (b) une structure non triviale $\overline{C}(L)$

3. $X \preceq Y$ *et* $Y \preceq Z \implies X \preceq Z$ *(transitivité)*

Démonstration. La propriété de reflexivité est triviale, car elle dépend de celle des éléments sur L. Concernant l'antisymétrie, il suffit de montrer par transitivité que la relation n'induit jamais de cycles (*c.f.* la définition 3.1.12).

$$X \preceq Y \implies \forall x \in X, \exists y \in Y, x \leq y \tag{5.72}$$
$$\iff X \leq Y \leq X \implies \exists x' \in X, y \leq x' \tag{5.73}$$

Or, $x \leq y \leq x'$. Il s'ensuit que $x = x'$ et $x = y$. Donc $X \subseteq Y$ et $X \supseteq Y$ implique $X = Y$. □

Le cas le plus simple est celui où L est totalement ordonné. Dans ce cas, $L \cong \overline{C}(L)$ puisque les éléments du treillis ne possèdent pas d'antichaînes. Ce n'est pas le cas des ordres partiels comme le montre la figure (5.6) où l'ensemble $\{2, 3\}$ représente donc la seule antichaîne, non ponctuelle, de la structure.

De plus, à partir du lemme (5.3.1), on déduit la propriété suivante :

Corollaire 5.3.1. *Soit un treillis atomistique* L, $(\overline{C}(\mathcal{J}(L)), \preceq) \cong (\mathcal{P}(\mathcal{J}(L)), \subseteq)$.

On peut également obtenir un résultat semblable sur les éléments co-premiers en renversant l'ordre de la relation sur les antichaînes :

$$X \succeq Y \iff \forall x \in X, \exists y, y \leq x \tag{5.74}$$

permettant de démontrer que $(\overline{C}(\mathcal{M}(L)), \succeq) \cong (\mathcal{P}(\mathcal{M}(L)), \supseteq)$.

Lemme 5.3.3. *Soit un ensemble $X \in \mathcal{P}(L)$ et $x \in X$, on note $\mathrm{Maj}(X)$, respectivement $\mathrm{Min}(X)$, les majorants, respectivement les minorants de X, on a les propriétés suivantes :*

1. $\mathrm{Maj}(X) \subset X, \exists x' \in \mathrm{Maj}(X)$ *tel que* $x \leq x'$ *;*
2. $\mathrm{Min}(X) \subset X, \exists x' \in \mathrm{Maj}(X)$ *tel que* $x' \leq x$ *;*
3. $\mathrm{Maj}(X)$ *et* $\mathrm{Min}(X)$ *sont des antichaînes ;*

Ce résultat est dû à l'application de la définition d'un majorant et d'un minorant dans un ensemble partiellement ordonné (*c.f.* 3.1.21). Par conséquent, on en déduit la relation suivante pour tout filtre et idéal principal de L : $\mathrm{Maj}(\downarrow x) = \{x\}$ et $\mathrm{Min}(\uparrow x) = \{x\}$.

Puisque la relation d'inclusion sur les filtres et idéaux étend naturellement celle du treillis, l'ordre sur les filtres et les idéaux est alors compatible avec celui sur les antichaînes.

Théorème 5.3.2. *Soit les filtres F, F' et les idéaux I, I', on a alors :*

$$F \subseteq F' \iff \mathrm{Min}(F) \preceq \mathrm{Min}(F') \tag{5.75}$$
$$I \subseteq I' \iff \mathrm{Maj}(I) \preceq \mathrm{Maj}(I') \tag{5.76}$$

Démonstration. De gauche à droite, la propriété découle de la définition de chacune des deux parties. Dans le sens inverse, il faut montrer que l'ordre sur deux antichaînes X, Y préserve l'ordre sur les filtres et les idéaux engendrés par X et Y :

$$X \preceq Y \implies \begin{cases} \downarrow X & \subseteq \downarrow Y \\ \uparrow X & \supseteq \uparrow Y \end{cases} \tag{5.77}$$

Posons $x \in \downarrow X$, il existe alors $x' \in \downarrow X$ tel que $x \leq x'$. De même, il existe un $y \in Y$ tel que $x' \leq y$ puisque $X \preceq Y$, donc $x \leq y$ et $x \in \downarrow Y$ et $\downarrow X \subseteq \downarrow Y$. La preuve est équivalente pour les filtres en utilisant la propriété de minimalité. □

En revanche, cela ne remet pas en cause le lemme (5.3.2), de telle sorte qu'on peut avoir deux ensembles X et Y avec $X \not\subseteq Y$ tels que $\downarrow X \subseteq \downarrow Y$ et $\uparrow X \supseteq \uparrow Y$.

Par exemple, sur la figure (5.7), pour les antichaînes 123 et 16, $\{1,2,3\} \not\subseteq \{1,6\}$. On a cependant :

$$\uparrow\{1,2,3\} = \{1,2,3,4,5,6\} \tag{5.78}$$
$$\uparrow\{1,6\} = \{1,4,5,6\} \tag{5.79}$$

tel que $\uparrow\{1,2,3\} \supset \uparrow\{1,6\}$. Puis :

$$\downarrow\{1,2,3\} = \{0,1,2,3\} \tag{5.80}$$
$$\downarrow\{1,6\} = \{0,1,2,3,6\} \tag{5.81}$$

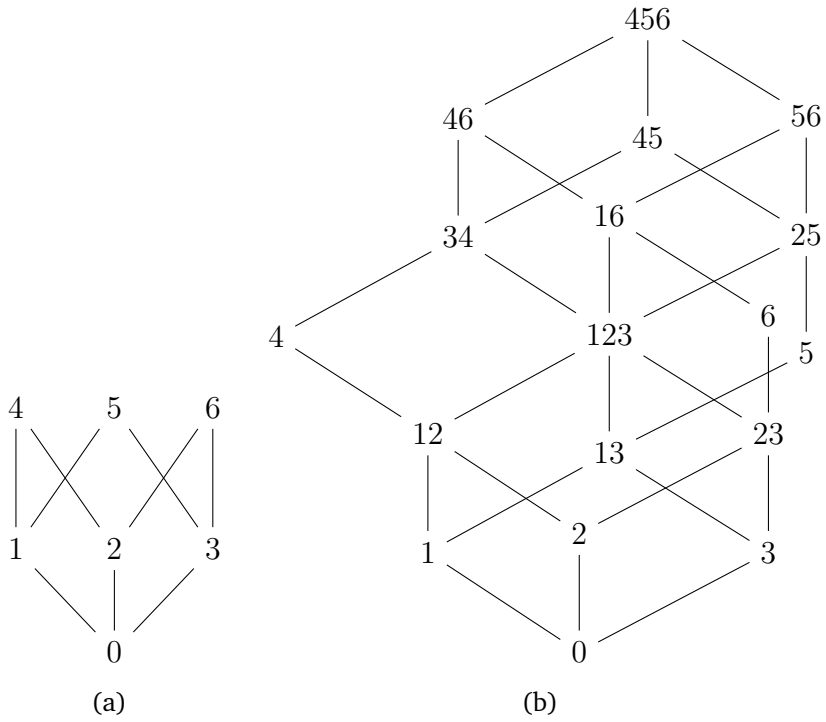


FIGURE 5.7 : (a) Le treillis des parties $\mathcal{P}(.)$ diminué d'un taquet (b) avec sa structure des antichaînes $\overline{C}(\mathcal{P}(.))$

tel que $\downarrow\{1,2,3\} \subset \downarrow\{1,6\}$.

On s'intéresse maintenant à la construction de points fixes dans la structure et à l'existence des opérateurs afférents.

Théorème 5.3.3. *Soit deux antichaînes X, Y, les bornes inférieure et supérieure sont définies de la manière suivante :*

$$\begin{cases} X \vee Y &= \mathrm{Maj}(X \cup Y) \\ X \wedge Y &= \mathrm{Maj}(\downarrow X \cap \downarrow Y) \end{cases} \tag{5.82}$$

Démonstration. Montrons en premier lieu que $\mathrm{Maj}(X \cup Y)$ est un majorant de $X \vee Y$. Posons $m \in X \cup Y$ tel que $m \in X$. Si $m \notin \mathrm{Maj}(X \cup Y)$, alors il existe $y \in X \cup Y$ tel que $z < y$ et $y \notin X$. Si $y \notin \mathrm{Maj}(X \cup Y)$ alors il existe $x \in X \cup Y$ tel que $m < y < x$. Or, $m, x \in X$, donc c'est impossible et m est un majorant.

On montre ensuite que $\mathrm{Maj}(X \cup Y)$ est la borne supérieure de X et Y. Soit une antichaîne Z telle que $X \preceq Z$ et $Y \preceq Z$ et $s \in \mathrm{Maj}(X \cup Y)$. Dans ce cas, il existe $m \in Z$ tel que $s \leq m$ et $\mathrm{Maj}(X \cup Y)$ est bien la borne supérieure.

Concernant la borne inférieure, montrons qu'elle existe pour tout couple d'antichaînes X et Y. Soit un $m \in \mathrm{Maj}(\downarrow X \cap \downarrow Y)$, par définition d'un minorant et de l'intersection ensembliste, $m \in \downarrow X$ et $m \in \downarrow Y$. Il existe donc $x \in X$ et $y \in Y$ tels que $m \leq x$ et $m \leq y$

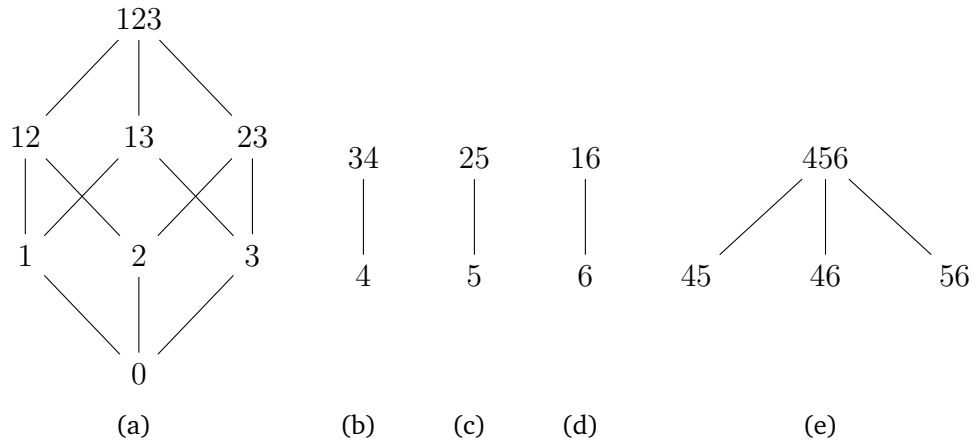


FIGURE 5.8 : Décomposition du treillis des antichaînes $\overline{C}(\mathcal{P}(.))$ de la figure 5.7

De la même manière que ci-dessus, on montre que $\mathrm{Maj}(\downarrow X \cap \downarrow Y)$ est la borne inférieure de X et Y. Soit une antichaîne Z telle que $Z \preceq X$ et $Z \preceq Y$, puisque les antichaînes préservent l'ordre sur les filtres, on a : $Z \subseteq \downarrow X$ et $Z \subseteq \downarrow Y$, alors $Z \subseteq \downarrow X \cap \downarrow Y$. On a donc $\mathrm{Maj}(Z) = Z$ et $Z \preceq \mathrm{Maj}(X \cap Y)$. □

Prenons, par exemple, les antichaînes $\{1, 2, 3\}$ et $\{6\}$, puisque 6 est un majorant de 1 et de 2 dans L, alors on a :

$$\{1, 2, 3\} \vee \{6\} = \{1, 6\}$$

A contrario, {2,3} est le majorant commun aux deux antichaînes tel que :

$$\{1, 2, 3\} \wedge \{6\} = \{2, 3\}$$

Comme on vient de le voir le treillis des antichaînes crée une relation sur des parties de $\mathcal{P}(L)$ telles que chaque partie ne contienne que des éléments orthogonaux.

En particulier, cette relation est trivialement monotone sur l'inclusion des filtres : les éléments de chaque partie génératrice G forment naturellement le treillis $\mathcal{P}(G)$. La figure 5.8 présente donc l'ensemble des décompositions du treillis des antichaînes de la figure 5.7.

Nous allons maintenant donner une définition des éléments maximaux de la structure.

Définition 5.3.3 (Antichaîne maximale)**.** *Soit une antichaîne $X \in \overline{C}(L)$, elle est dite maximale si et seulement si, elle vérifie la propriété suivante :*

$$\uparrow \mathrm{Min}(X) \uplus \downarrow \mathrm{Maj}(X) = L \tag{5.83}$$
$$\uparrow \mathrm{Min}(X) \cap \downarrow \mathrm{Maj}(X) = \varnothing \tag{5.84}$$

Le treillis des antichaînes maximales $\overline{C}_m(L)$ est représenté par la figure (5.9).

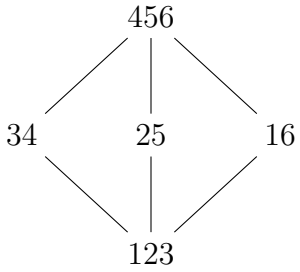


FIGURE 5.9 : Treillis des antichaînes maximales de la figure 5.7

Les taquets représentent donc les ensembles $\mathcal{J}(L)$ et $\mathcal{M}(L)$. En particulier, dans un treillis complet, toutes les antichaînes possèdent nécessairement une antichaîne maximale [115].

Pour montrer comment elles sont construites, on va étudier le rapport entre les filtres/idéaux et les perspectives.

5.3.2 Relations entre les antichaînes et les paires filtre-idéal

Nous introduisons les variantes irréflexives des relations de perspectives $\not\nearrow$ et $\not\swarrow$, soient :

$$\begin{cases} \not\nearrow^{=} j & \triangleq \{x \mid j \not< x\} \\ \not\swarrow^{=} m & \triangleq \{x \mid x \not< m\} \end{cases} \tag{5.85}$$

En particulier, on a toujours $j \in \not\nearrow^{=} j$ et $m \in \not\swarrow^{=} m$. On en déduit la propriété suivante :

Lemme 5.3.4. *Soit un ensemble $X \in \mathcal{P}(L)$, on observe les relations suivantes :*

$$\not\nearrow^{=} X = \not\nearrow X \cup \mathrm{Min}(X) \tag{5.86}$$
$$\not\swarrow^{=} X = \not\swarrow X \cup \mathrm{Maj}(X) \tag{5.87}$$

La propriété découle directement de la définition. Les variantes irréflexives préservent donc la base des parties qu'elles engendrent, autrement dit des antichaînes.

Par exemple, sur le treillis de la figure (5.10), on a :

$$\begin{aligned} \not\swarrow\{e\} &= \{c, d, f, g, h, i, j, k, l, m, n, 1\} \\ \not\swarrow^{=}\{e\} &= \{c, d, e, f, g, h, i, j, k, l, m, n, 1\} \\ &= \{e\} + \not\swarrow\{e\} \end{aligned}$$

Lemme 5.3.5. *Soit un ensemble X dans $\mathcal{P}(L)$, on a :*

$$(\not\swarrow \circ \not\nearrow)\,\mathrm{Min}(X) = F \tag{5.88}$$
$$(\not\nearrow \circ \not\swarrow)\,\mathrm{Maj}(X) = I \tag{5.89}$$
$$\tag{5.90}$$

tel que F soit un filtre et I un idéal.

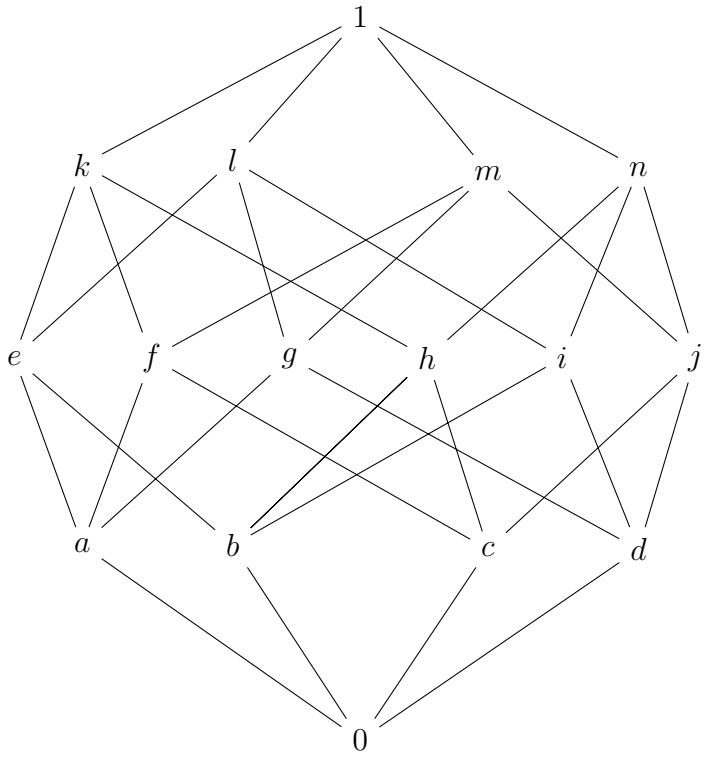


FIGURE 5.10 : Le treillis des parties $\mathcal{P}(.)$ à quatre éléments

Démonstration. C'est un corollaire de la définition des opérations $\nearrow$ et $\swarrow$ et du théorème (2.3.1). Toute partie d'un ensemble ordonné peut être fermée pour l'ordre et ses bornes sont nécessairement des générateurs. □

Suivant le même treillis que dans l'exemple précédent, posons $X = \{a, f\}$, on a alors :

$$\begin{aligned} \nearrow X &= \{0, b, c, d, h, i, j, n\} \\ \swarrow\{0, b, c, d, h, i, j, n\} &= \{a, e, f, g, k, l, m, 1\} \\ &= \uparrow a \end{aligned}$$

Inversement, on aura :

$$\begin{aligned} \swarrow X &= \{b, d, e, g, h, i, j, k, l, m, n, 1\} \\ \nearrow\{b, d, e, g, h, i, j, k, l, m, n, 1\} &= \{0, a, c, f\} \\ &= \downarrow f \end{aligned}$$

On note également que cela fonctionne pour les cônes d'où la caractérisation par le biais des minorants et majorants. Étant donné que les flèches – barrées ou non – sont des opérations ensemblistes, ce lemme se réécrit de la manière suivante.

Théorème 5.3.4. *Soit un treillis complet L. On a les propriétés de fermeture suivantes*

dans $(\mathcal{P}(L), \subseteq)$ *:*

$$\uparrow X = (\not\swarrow \circ \not\nearrow)Y \tag{5.91}$$
$$\downarrow X = (\not\nearrow \circ \not\swarrow)Y \tag{5.92}$$

où $Y \subseteq X$.

En particulier, pour n'importe quel élément ponctuel, on engendre soit le filtre principal, soit l'idéal principal. Dans le cas des perspectives irréflexives, étant donné qu'elles préservent, soit le minorant, soit le majorant, la composition donne un point de vue intéressant.

Lemme 5.3.6. *Soit un treillis complet* L*, pour tout* $x \in L$*, on a :*

$$x \in (\not\swarrow^{=} \circ \not\nearrow^{=})\{x\} \tag{5.93}$$
$$x \in (\not\nearrow^{=} \circ \not\swarrow^{=})\{x\} \tag{5.94}$$

Démonstration. Les démonstrations de chaque propriété sont duales, on se contente donc de démontrer la première. On note $\overline{F}$, l'ensemble engendré par $\not\nearrow^{=}\{x\}$, on sait que $x \in \overline{F}$, puisque x est le seul minorant de l'ensemble. Puis, étant donné que x n'est pas en relation avec les éléments de $\not\swarrow \overline{F}$, il est nécessairement un majorant de $\not\swarrow^{=} \overline{F}$ et appartient donc au résultat. □

Corollaire 5.3.2. *Pour tout* $x \in L$*, les ensembles suivants :*

$$\{x\} \cup \mathrm{Min}(\not\swarrow^{=} \circ \not\nearrow^{=})\{x\} \tag{5.95}$$
$$\{x\} \cup \mathrm{Maj}(\not\nearrow^{=} \circ \not\swarrow^{=})\{x\} \tag{5.96}$$

sont des antichaînes.

Corollaire 5.3.3. *Pour toute antichaîne* X *d'un treillis* L*, les propriétés suivantes sont vérifiées :*

$$\not\nearrow^{=} X = \downarrow Y \tag{5.97}$$
$$\not\swarrow^{=} X = \uparrow Y \tag{5.98}$$

telles que $X \subseteq Y \in \overline{C}(L)$.

Étant donné que chaque perspective construit tour à tour un filtre puis un idéal, et réciproquement, une antichaîne est donc maximale si elle ne peut être complétée.

Théorème 5.3.5. *Une antichaîne* X *est maximale si et seulement si :*

$$\mathrm{Maj}(\not\nearrow^{=} \circ \not\swarrow^{=})X = X \tag{5.99}$$
$$\mathrm{Min}(\not\swarrow^{=} \circ \not\nearrow^{=})X = X \tag{5.100}$$

Chaque antichaîne X ne peut donc être plus grande pour l'inclusion ensembliste.

Soit, par exemple, l'antichaîne $X = \{a, h, i, j\}$. Vérifions qu'elle est maximale[5] :

$$\begin{aligned} \not\uparrow^{=} \circ \not\downarrow^{=} X &= \not\uparrow^{=} \{\underline{a}, e, f, g, \underline{h, i, j}, k, l, m, n, 1\} \\ &= \not\uparrow^{=} \circ \uparrow \{\underline{a, h, i, j}\} \\ &= \{0, \overline{a}, b, c, d, \overline{h, i, j}\} \\ &= \downarrow \{\overline{a, h, i, j}\} \end{aligned}$$

dont les majorants sont donc $\{a, h, i, j\}$. Maintenant, étant donnée l'antichaîne $X = \{a, n\}$, on a :

$$\begin{aligned} \not\downarrow^{=} \circ \not\uparrow^{=} X &= \not\downarrow^{=} \{0, \overline{a}, b, c, d, h, i, j, \overline{n}\} \\ &= \not\downarrow^{=} \circ \downarrow \{\overline{a, n}\} \\ &= \{\underline{a}, e, f, g, k, l, m, \underline{n}, 1\} \\ &= \uparrow \{\underline{a, n}\} \end{aligned}$$

dont les minorants sont donc $\{a, n\}$.

En particulier, les paires maximales dans $\overline{C}_m(L)$ sont compatibles avec la relation d'ordre $(\preceq)$.

Théorème 5.3.6. *Soit deux antichaînes maximales X et Y. On a l'identité suivante :*

$$X \preceq Y \iff \not\uparrow(X \cap Y) = \downarrow \mathrm{Maj}(X \Delta Y) \tag{5.101}$$

Démonstration. Par définition, tous les éléments de Y sont des majorants des éléments de X. Par ailleurs, il existe au moins un $x \in X$ et un $y \in Y$ tels que $x < y$, sinon $X \cup Y$ serait une antichaîne ce qui contredit la maximalité.

L'intersection $X \cap Y$ contient donc les x's qui ne sont pas majorées dans Y. Puisque $X \Delta Y = (X \cup Y) - (X \cap Y)$, la différence engendre un ordre partiel avec un ou plusieurs majorants. Donc, $X \cap Y$ est une antichaîne non maximale qui peut être complétée par les majorants de l'idéal engendré par $X \cap Y$. □

Reprenons, par exemple, les antichaînes maximales $X = \{a, h, i, j\}$ et $Y = \{a, n\}$. Le graphe de la figure (5.11) illustre le fait que n est un majorant de $\{h, i, j\}$ pour la relation $(\leq)$. L'intersection contient donc la partie de chaque antichaîne qui est à la fois un minorant et majorant, on a donc $X \cap Y = \{a\}$. Les éléments dans $X \Delta Y$ sont donc nécessairement orthogonaux à $X \cap Y$ et on a :

$$\begin{aligned} \not\uparrow(X \cap Y) &= \{0, b, c, d, h, i, j, n\} \\ &= \downarrow \{h, i, j, n\} \\ &= \downarrow (\{a, h, i, j\} \Delta \{a, n\}) \\ &= \downarrow \{n\} \\ &= \downarrow \mathrm{Maj}(X \Delta Y) \end{aligned}$$

[5] Chaque élément surmonté d'un trait suscrit désigne un majorant de l'ensemble. À l'inverse, chaque élément souligné désigne un minorant de celui-ci.

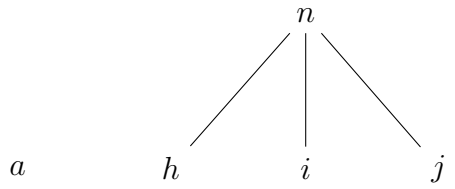


FIGURE 5.11 : Graphe de la relation $(\preceq)$ des antichaînes $\{a, h, i, j\}$ et $\{a, n\}$

5.3.3 Représentation générale des treillis

Nous allons maintenant présenter une application de la méthodologie d'Urquhart [133] pour la représentation générale des treillis, c.-à-d. ne faisant pas l'hypothèse de la distributivité entre les opérateurs $(\wedge)$ et $(\vee)$; le but étant de définir une opération de négation dans le treillis des partitions Π^Ω.

Celle-ci va nous permettre d'étudier l'extension canonique associant chaque partition du treillis à un sous-ensemble de paires filtre-idéal, représentant canoniquement les partitions pouvant être déduites ou niées à partir de celle-ci. On va donc construire un espace sur $\mathcal{J}(\Pi^\Omega) \times \mathcal{M}(\Pi^\Omega)$.

Définition 5.3.4 (Paire filtre-idéal)**.** *Soit un treillis L, une paire filtre-idéal est un élément $x \in \mathcal{J}(L) \times \mathcal{M}(L)$ où $x = (x_1, x_2)$. x_1 et x_2 sont respectivement un filtre et un idéal, tandis que $x_1 \cap x_2 = \varnothing$. Une paire n'est pas nécessairement exhaustive telle que $x_1 \cup x_2 = L$.*

Par exemple, la paire $(\uparrow 12|3|4, \downarrow 1|234)$ est telle que :

$$\uparrow 12|3|4 \cup \downarrow 1|234 \subsetneq \Pi^\Omega$$

En effet, la partition $13|24$ n'appartient ni à l'un ni à l'autre ensemble. En revanche, on remarque que :

$$\begin{cases} 13|24 & \in \not\uparrow 12|3|4 \\ \downarrow 1|234 & \subset \not\uparrow 12|3|4 \end{cases}$$

Donc la partition $13|24$ nie $12|34$ mais est indépendante de $1|234$. Ce qui va nous permettre de définir une relation sur les paires qui étend celle sur les parties de Π^Ω.

Définition 5.3.5. *Soit les paires filtre-idéal x, y, on définit la relation d'inclusion suivante :*

$$x \leq y \iff \begin{cases} x_1 \subseteq y_1 \\ x_2 \subseteq y_2 \end{cases} \tag{5.102}$$

Par exemple, étant données les partitions $R = 12|3|4$, $S = 123|4$, $P = 1|234$ et $Q = 1|24|3$, on a donc :

$$P \wedge R = \bot$$
$$Q \wedge S = \bot$$

Donc,

$$\uparrow P \subset \uparrow Q, \downarrow R \subset \downarrow S \implies (\uparrow P, \downarrow R) < (\uparrow Q, \downarrow S)$$

On notera $x \leq_1 y$ lorsque $x_1 \subseteq y_1$, et $x \leq_2 y$ lorsque $x_2 \subseteq y_2$.

Définition 5.3.6 (Paire filtre-idéal maximale)**.** *Soit* $(\mathcal{FI}_m(\Pi^\Omega), \leq_1, \leq_2)$*, l'ensemble des paires filtre-idéal maximales sur* $\mathcal{J}(\Pi^\Omega) \times \mathcal{M}(\Pi^\Omega)$ *telle que, pour tout* $x, y \in \mathcal{FI}_m$*, on a :*

$$x \leq_1 y \wedge x \leq_2 y \implies x = y \tag{5.103}$$

$(\mathcal{FI}_m(\Pi^\Omega), \leq_1, \leq_2)$ *est un ensemble doublement ordonné. On munit la structure*$(\mathcal{FI}_m(\Pi^\Omega)$ *des projections suivantes :*

$$\pi_1 : \mathcal{FI}_m(\Pi^\Omega) \to \mathcal{P}(\mathcal{F}(\Pi^\Omega)) \tag{5.104}$$
$$\pi_2 : \mathcal{FI}_m(\Pi^\Omega) \to \mathcal{P}(\mathcal{I}(\Pi^\Omega)) \tag{5.105}$$

telles que :

$$\pi_1(x) = \{F \in \mathcal{F}(\Pi^\Omega) \mid (F, x_2) \in \mathcal{FI}_m(\Pi^\Omega)\} \tag{5.106}$$
$$\pi_2(x) = \{I \in \mathcal{I}(\Pi^\Omega) \mid (x_1, I) \in \mathcal{FI}_m(\Pi^\Omega)\} \tag{5.107}$$

La paire $(\uparrow Q, \downarrow S)$, définie ci-dessus, est par conséquent maximale tandis que la paire $(\uparrow P, \downarrow R) \notin \mathcal{FI}_m(\Pi^\Omega)$. On aura également les ensembles images suivants :

$$\leq_1 (x) = \{y \in \mathcal{FI}_m(\Pi^\Omega) \mid x \leq_1 y\} \tag{5.108}$$
$$\leq_2 (x) = \{y \in \mathcal{FI}_m(\Pi^\Omega) \mid x \leq_2 y\} \tag{5.109}$$

$\leq_1 (x)$ construit donc un ensemble de filtres dont l'union $\bigcup \leq_1 (x)$ est le filtre maximal contenant x ; la même propriété étant dualement obtenue pour l'union des idéaux. Chaque ensemble obtenu est donc une séquence maximale pour l'inclusion garantissant la disjonction des filtres et des idéaux maximaux bien que ceux-ci ne soient pas nécessairement premiers (*c.f.* lemme 3.3.1).

Chaque partition du treillis Π^Ω peut alors être plongée dans l'espace des paires de la manière suivante.

Définition 5.3.7 (Modèle d'interprétation)**.** *Soit l'homomorphisme* $m : \Pi^\Omega \to \mathcal{P}(\mathcal{FI}_m(\Pi^\Omega))$ *défini par :*

$$m(P) = \{x \in \mathcal{FI}_m(\Pi^\Omega) \mid P \in \pi_1(x)\} \tag{5.110}$$

m *construit alors l'ensemble des contextes, c.-à-d. les filtres maximaux où* P *est vrai. On peut alors écrire :*

$$x \models_1 P \iff x \in m(P) \tag{5.111}$$

$(\mathcal{FI}_m(\Pi^\Omega), \leq_1, \leq_2, m)$ *désigne alors les modèles d'interprétation – structures de Kripke – où les paires sont les mondes dans lequels sont interprétables chaque partition à travers* m*.*

Lemme 5.3.7. *Soit une paire* $x \in \mathcal{FI}_m(\Pi^\Omega)$*, il existe au moins une paire* y *telle que :*

$$(x_1 = y_1) \text{ et } (x_2 \neq y_2) \tag{5.112}$$
$$\text{ou } (x_1 \neq y_1) \text{ et } (x_2 = y_2) \tag{5.113}$$

Démonstration. Il suffit de constater que pour tout filtre $\uparrow P$, la perspective $\not\uparrow P$ n'est pas un idéal de l'ordre mais un cône négatif dont chaque majorant forme un idéal compatible avec $\uparrow P$. □

Par exemple, étant donnée la partition $P = 12|3|4$, on a le résultat suivant :

$$\mathrm{Maj}(\not\uparrow P) = \{134|2, 1|234, 13|24, 14|23\}$$

telle que pour tout $Q \in \mathrm{Maj}(\not\uparrow P)$:

$$\begin{cases} \uparrow P \cap \downarrow Q & = \varnothing \\ \uparrow P \cup \downarrow Q & \subset \Pi^{\Omega} \end{cases}$$

Chaque paire dans $\mathcal{FI}_m$ permet donc d'évaluer un élément du treillis dans le contexte formé par le filtre, représentant ce qui est démontrable, et de l'idéal, représentant ce qui est réfutable. On peut donc réaliser la liaison entre ces paires et les perspectives.

Corollaire 5.3.4. *La structure des paires $\mathcal{FI}_m(\Pi^{\Omega})$ est telle que pour tout x :*

$$|\pi_1(x)| > 1 \tag{5.114}$$
$$|\pi_2(x)| > 1 \tag{5.115}$$

De plus, étant donné que ces paires sont définies à l'aide de filtres et idéaux principaux, ils forment donc nécessairement des antichaînes.

Théorème 5.3.7. *Soit une paire filtre-idéal $(\uparrow P, \downarrow Q)$, celle-ci est maximale si il existe une antichaîne X telle que $\{P, Q\} \subseteq X$ et X soit maximale pour l'ordre dans le treillis $\overline{C}_m(\Pi^{\Omega}) - \{\bot, \top\}$.*

Démonstration. On va supposer que X n'est pas maximale pour l'ordre. Posons les antichaînes maximales $X = \{P, Q, R\}$ et $Y = \{P, Q'\}$, cela implique que Q' est un majorant de Q et R tel que $\downarrow Q \subset \downarrow Q'$. Or, $P \leq_1 P$ et $Q \leq_2 Q' \implies (P, Q) = (P, Q')$. Donc la paire $(\uparrow P, \downarrow Q)$ est maximale si et seulement si $Y = X$. □

Corollaire 5.3.5. *Une paire filtre-idéal (x_1, x_2) est exhaustive si et seulement si $\{\mathrm{Min}(x_1) \cup \mathrm{Maj}(x_2)\}$ est une antichaîne maximale.*

La preuve est évidente et est également une conséquence de la définition (5.3.3) et par le corollaire (5.3.4), de telles paires n'existent pas. On peut donc formuler l'interprétation des partitions vis-à-vis d'une paire particulière.

Définition 5.3.8 (Sémantique des modèles des partitions)**.** *Soit une paire maximale $x = (x_1, x_2) \in \mathcal{FI}_m(\Pi^{\Omega})$, pour toute partition P, une seule des propriétés suivantes est vérifiée :*

1. $P \in x_1, P \notin x_2$

2. $P \notin x_1, P \in x_2$
3. $P \notin x_1, P \notin x_2$

*Le dernier cas étant la négation des deux autres cas soulignant l'*incomplétude *du modèle construit sur x tel que :*

$$x \models_1 P \implies x \not\models_0 P \tag{5.116}$$
$$x \models_0 P \implies x \not\models_1 P \tag{5.117}$$
$$x \not\models_1 P, x \not\models_0 P \implies x \models_\varnothing P \tag{5.118}$$

En particulier, $x \models_\varnothing P$ n'implique donc pas, ni $x \models_1 P$, ni $x \models_0 P$.

Par exemple, on a :

$$(\uparrow 12|3|4, \downarrow 1|234) \models_\varnothing 13|24$$

puisque $13|24 \notin \uparrow 12|3|4 \cup \downarrow 1|234$.

Réciproquement, le théorème (5.3.6) et l'exemple précédent permettent de définir des paires filtre-idéal maximales depuis une antichaîne maximale.

Théorème 5.3.8. *Soient deux antichaînes X et Y dans $\overline{C}_m(\Pi^\Omega) - \{\bot, \top\}$ telles que $X \preceq Y$, on pose :*

$$J = \mathcal{J}_{\leq X \cap Y} \tag{5.119}$$
$$M = \mathcal{M}_{> X \Delta Y} \tag{5.120}$$

Alors, pour tout $(j, m) \in J \times M$, (j, m) est une paire filtre-idéal maximale.

Démonstration. Par définition, le cône positif $\uparrow(X \cap Y)$ est disjoint du cône négatif $\downarrow \mathrm{Maj}(X \Delta Y)$. Pour tout $j \in J$, $\uparrow(X \cap Y) \leq_1 \uparrow j$ et par construction, tout $m \in M$ est maximal pour l'ordre $(\leq_2)$ donc les paires (j, m) sont nécessairement maximales. □

Corollaire 5.3.6. *Soit une partition P, on a l'identité suivante sur ses modèles maximaux :*

$$m(P) = \mathcal{J}_{\leq P} \times \mathcal{M}_{\not\geq P} \tag{5.121}$$

C'est la conséquence directe du théorème précédent et indique que l'extension canonique de Π^Ω est équivalente à sélectionner les modèles maximaux pour chaque partition (*c.f.* déf. 3.3.4).

Définition 5.3.9. *Soit un ensemble de paires maximales $A \subseteq \mathcal{FI}_m$, A est $\leq_1$-croissant si et seulement si :*

$$x \in A \implies \forall y, x \leq_1 y, y \in A \tag{5.122}$$

A est $\leq_2$-croissant si et seulement si :

$$x \in A \implies \forall y, x \leq_2 y, y \in A \tag{5.123}$$

Les ensembles renvoyés π_1 et π_2 sont naturellement, respectivement, $\leq_1$-croissant et $\leq_2$-croissant.

La structure $\mathcal{FI}_m(\Pi^\Omega)$ est donc le modèle algébrique d'un système déductif trivalué pour lequel on peut définir la valuation $v : \Pi^\Omega \times \mathcal{FI}_m(\Pi^\Omega) \to \{\varnothing, 0, 1, \{0,1\}\}$ avec :

$$v(P,x) = \begin{cases} 1 & \iff x \models_1 P \\ 0 & \iff x \models_0 P \\ \varnothing & \iff x \models_\varnothing P \end{cases} \tag{5.124}$$

où les valeurs dans $\{\varnothing, 0, 1\}$ représentent les éléments désignés, permettant d'évaluer la véracité d'une expression dans Π^Ω vis-à-vis de ses modèles $\mathcal{FI}_m(\Pi^\Omega)$ [7, 15, 85].

La valeur $\varnothing$ représente donc l'indétermination d'une partition par un modèle particulier : lorsque $v(P,x) = \varnothing$, P peut être ni vraie ni fausse selon x.

Nous allons maintenant définir les cartes associées à la structure $(\mathcal{FI}_m, \leq_1, \leq_2)$ et permettant de manipuler les modèles.

Définition 5.3.10. *Soit l'ensemble $A \subseteq \mathcal{FI}_m(\Pi^\Omega)$, on définit les cartes $l, r : \mathcal{P}(\mathcal{FI}_m(\Pi^\Omega)) \to \mathcal{P}(\mathcal{FI}_m(\Pi^\Omega))$ par :*

$$l(A) = \{x \in \mathcal{FI}_m(\Pi^\Omega) \mid \forall y \in \mathcal{FI}_m(\Pi^\Omega), x \leq_1 y \implies y \notin A\} \tag{5.125}$$

$$r(A) = \{x \in \mathcal{FI}_m(\Pi^\Omega) \mid \forall y \in \mathcal{FI}_m(\Pi^\Omega), x \leq_2 y \implies y \notin A\} \tag{5.126}$$

ou de manière équivalente :

$$l(A) = \{x \in \mathcal{FI}_m(\Pi^\Omega) \mid \leq_1 (x) \subseteq \mathcal{FI}_m(\Pi^\Omega) - A\} \tag{5.127}$$

$$r(A) = \{x \in \mathcal{FI}_m(\Pi^\Omega) \mid \leq_2 (x) \subseteq \mathcal{FI}_m(\Pi^\Omega) - A\} \tag{5.128}$$

Étant donné un ensemble de paires, chaque carte construit alternativement l'ensemble complémentaire des paires tel que ces derniers soient disjoints selon l'inclusion sur les filtres ou les idéaux, et sont donc semblables aux perspectives.

Selon le treillis des antichaînes maximales de la figure (5.9), on a les paires maximales $\mathcal{FI}_m(L) = \{(\uparrow 1, \downarrow 6), (\uparrow 2, \downarrow 5), (\uparrow 3, \downarrow 4)\}$. Posons $A = \{(\uparrow 1, \downarrow 6)\}$, on a alors :

$$r(A) = \{(\uparrow 2, \downarrow 5), (\uparrow 3, \downarrow 4)\}$$

correspondant aux paires dont l'idéal principal est engendré par un élément contenu dans :

$$\mathrm{Maj}(\not\uparrow 6) = \mathrm{Maj}(\uparrow 1) = \{4, 5\}$$

et donc complémentaire de 6. Tandis que l'application de la carte $l(.)$ renvoie les paires dont les filtres ne sont pas dans le cône positif $\uparrow\{2,3\}$ d'où :

$$\mathrm{Min}(\not\downarrow\{2,3\} = \mathrm{Min}(\downarrow\{4,5\}) = \{1\}$$

En particulier, $A = l \circ r(A) = \{(1,6)\}$. On remarque que la composition de ces opérations est similaire au calcul du complémentaire de $\pi_2(.)$ puis de $\pi_1(.)$ sur A, d'où le prochain résultat.

Lemme 5.3.8. *Les cartes l et r admettent les définitions suivantes :*

$$l(A) = \{x \mid x_1 \notin \bigcup_{y \in A} \pi_1(y)\} \tag{5.129}$$

$$r(A) = \{x \mid x_2 \notin \bigcup_{y \in A} \pi_2(y)\} \tag{5.130}$$

Posons la partition $P = 123|4$ et $A \triangleq m(P)$ tel que :

$$\begin{aligned} A = \quad & (\uparrow 12|3|4 \times \downarrow \mathcal{M}_{\not\geq 12|3|4}) \\ \cup \quad & (\uparrow 13|2|4 \times \downarrow \mathcal{M}_{\not\geq 13|2|4}) \\ \cup \quad & (\uparrow 1|23|4 \times \downarrow \mathcal{M}_{\not\geq 1|23|4}) \end{aligned}$$

$r(A)$ renvoie les paires dont les idéaux sont disjoints de toutes les paires de A, soit $\mathcal{M}(\Pi^\Omega) - \bigcup_{y \in A} \pi_2(y)$, d'où :

$$\begin{aligned} r(A) \quad &= \uparrow \mathcal{J}_{\not\leq 123|4} \times \downarrow 123|4 \\ &= \{\uparrow 14|2|3, \uparrow 1|24|3, \uparrow 1|34|2\} \times \downarrow 123|4 \end{aligned}$$

On remarque que $r(A)$ construit le cône négatif ou l'union des idéaux qui sont disjoints de ceux de A, par conséquent toutes les paires compatibles sont celles contenant l'idéal $\downarrow 123|4$. La composition avec $l(.)$ va donc renvoyer toutes les paires dont les filtres sont disjoints, soit $\uparrow(\mathcal{J}(\Pi^\Omega) - \mathcal{J}_{\not\leq 123|4})$, telle que :

$$l \circ r(A) = \uparrow \mathcal{J}_{\leq 123|4} \times \downarrow \mathcal{M}_{\not\geq 123|4} = A$$

La composition de ces cartes joue donc le même rôle que les perspectives en calculant successivement,

1. les idéaux maximaux disjoints de la perspective $\not\downarrow(\pi_2[A])$ pour $r(.)$;
2. les filtres maximaux disjoints de la perspective $\not\uparrow(\pi_1[A])$ pour $l(.)$;

On a vu précédemment que lorsqu'on préserve les majorants et minorants dans les applications successives des perspectives, on obtient alors une antichaîne maximale. On en déduit que $(l \circ r)(A)$ construit le modèle maximal hérité de l'ensemble des paires A.

En utilisant le théorème (5.3.7) et en étendant son corollaire (5.3.5) au cas d'une collection de paires, on a le résultat suivant.

Théorème 5.3.9. *Soit une partition P telle que $A = m(P)$, on a :*

$$(l \circ r)(A) = A \tag{5.131}$$

A est alors un élément l-stable, soit un point fixe de l'opération de fermeture induite par la composition des deux cartes.

En effet, par construction, le modèle associé à P est formé depuis une antichaîne maximale dont les filtres sont engendrés par ses minorants et les idéaux sont engendrés par ses majorants.

Définition 5.3.11. *On note $L_m(\Pi^\Omega)$, la famille des ensembles l-stables dans $\mathcal{P}(\mathcal{FI}_m)$. Soit le treillis complet $(L_m(\Pi^\Omega), \vee, \wedge, \bot, \top)$ où $\bot = \varnothing$ et $\top = \mathcal{FI}_m$. Pour tout $A, B \in L_m(\Pi^\Omega)$, on a les opérations suivantes :*

$$\begin{cases} A \wedge B & = A \cap B \\ A \vee B & \triangleq l(r(A) \cap r(B)) \end{cases} \tag{5.132}$$

Ces cartes jouent donc le rôle de deux opérations de négation. Posons une partition P, soient les modèles suivants :

$$m_1(P) = \{x \in \mathcal{FI}_m(\Pi^\Omega) \mid x \models_1 P\} \tag{5.133}$$
$$m_0(P) = \{x \in \mathcal{FI}_m(\Pi^\Omega) \mid x \models_0 P\} \tag{5.134}$$

alors on a :

$$m_0(P) = r(m_1(P)) \tag{5.135}$$
$$m_1(P) = l(m_0(P)) \tag{5.136}$$

Dans les faits, à l'aide de ses opérateurs, la sémantique est en réalité celle d'une logique bivaluée. Posons $A = m(12|3|4)$ et $B = m(13|2|4)$ et calculons $A \vee B$. On a :

$$\begin{aligned} A &= \uparrow 12|3|4 \times (\downarrow 134|2, \downarrow 1|234, \downarrow 13|24, \downarrow 14|23) \\ B &= \uparrow 13|2|4 \times (\downarrow 124|3, \downarrow 1|234, \downarrow 12|34, \downarrow 14|23) \end{aligned}$$

on obtient alors :

$$\begin{aligned} r(A) = \; & ((\uparrow 14|2|3, \uparrow 1|24|3, \uparrow 1|2|34) \times \downarrow 123|4) \\ \cup \; & ((\uparrow 13|2|4, \uparrow 1|23|4, \uparrow 1|2|34) \times \downarrow 124|3) \\ \cup \; & ((\uparrow 13|2|4, \uparrow 14|2|3, \uparrow 1|23|4, \uparrow 1|24|3) \times \downarrow 12|34) \end{aligned}$$

et

$$\begin{aligned} r(B) = \; & ((\uparrow 14|2|3, \uparrow 1|24|3, \uparrow 1|2|34) \times \downarrow 123|4) \\ \cup \; & ((\uparrow 12|3|4, \uparrow 1|23|4, \uparrow 1|24|3) \times \downarrow 134|2) \\ \cup \; & ((\uparrow 12|3|4, \uparrow 14|2|3, \uparrow 1|23|4, \uparrow 1|2|34) \times \downarrow 13|24) \end{aligned}$$

On remarque aisément que les idéaux dans $\pi_2(r(a))$ et $\pi_2(r(b))$ sont les majorants accessibles depuis $\uparrow \pi_1(A)$ et $\uparrow \pi_1(B)$. En particulier, l'intersection vaut :

$$r(A) \cap r(B) = (\uparrow 14|2|3, \uparrow 1|24|3, \uparrow 1|2|34) \times \downarrow 123$$

qui représente donc la négation du modèle maximal accessible depuis A et B et on a :

$$l(r(A) \cap r(B)) = \uparrow 123 \times \downarrow (\mathcal{M}(\Pi^\Omega) - 123)$$

Comme attendu, la partition résultante de $12|3|4 \vee 13|2|4$ est $123|4$.

Posons maintenant $A = m(12|3|4)$ et $B = m(1|2|34)$, on a :

$$\begin{aligned} A &= \uparrow 12|3|4 \times (\downarrow 134, \downarrow 234, \downarrow 13|24, \downarrow 14|23) \\ B &= \uparrow 1|2|34 \times (\downarrow 123, \downarrow 124, \downarrow 13|24, \downarrow 14|23) \end{aligned}$$

on obtient :

$$\begin{aligned} r(A) = \; & ((\uparrow 14|2|3, \uparrow 1|24|3, \uparrow 1|2|34) \times \downarrow 123) \\ \cup \; & ((\uparrow 13|2|4, \uparrow 1|23|4, \uparrow 1|2|34) \times \downarrow 124) \\ \cup \; & ((\uparrow 13|2|4, \uparrow 14|2|3, \uparrow 1|23|4, \uparrow 1|24|3) \times \downarrow 12|34) \end{aligned}$$

et

$$\begin{aligned} r(B) = \; & ((\uparrow 12|3|4, \uparrow 1|23|4, \uparrow 1|24|3) \times \downarrow 134) \\ \cup \; & ((\uparrow 12|3|4, \uparrow 13|2|4, \uparrow 14|2|3) \times \downarrow 234) \\ \cup \; & ((\uparrow 13|2|4, \uparrow 14|2|3, \uparrow 1|23|4, \uparrow 1|24|3) \times \downarrow 12|34) \end{aligned}$$

$A \vee B = 12|34$ puisque :

$$\begin{aligned} l(r(A) \cap r(B)) = \; & (\uparrow 12|3|4 \times (\downarrow 134, \downarrow 234, \downarrow 13|24, \downarrow 14|23)) \\ \cup \; & (\uparrow 1|2|34 \times (\downarrow 123, \downarrow 124, \downarrow 13|24, \downarrow 14|23)) \end{aligned}$$

qui s'obtient directement par $l(r(A \cup B))$. Ce qui mène au résultat suivant :

Lemme 5.3.9. *Soit deux partitions $P, Q \in \Pi^\Omega$ telles que $A = m(P)$ et $B = m(Q)$, on a l'identité suivante :*

$$r(A \cup B) = r(A) \cap r(B) \iff P \vee Q = \phi(P) \cup \phi(Q) \tag{5.137}$$

Cette identité indique donc que la fermeture transitive n'est pas nécessaire dans l'obtention des classes finales et est similaire à l'opération d'union dans le treillis $(\mathcal{P}(P \cup Q), \cup, \cap)$.

5.4 Conclusion

Dans ce chapitre, nous avons développé une approche constructiviste dans le but de définir de nouveaux opérateurs qui s'accordent avec les opérateurs algébriques naturels du treillis des partitions, pour lesquels, il n'existe pas de cadre formel adéquat décrivant le comportement attendu lorsqu'on les combine, ni pour l'interprétation de la dualité les connectant.

Nous avons également rappelé l'intérêt de la formulation algébrique du problème de la partition médiane. Dans ce contexte, nous avons décrit un cadre formel indépendant de la propriété de distributivité, qui est habituellement au cœur des problématiques de modélisation à partir de l'algèbre de Lindembaum-Tarski.

La recherche d'une partition réalisant un compromis au sein d'un ensemble de partitions $\mathbf{P}$ peut donc être formulée par la quantification d'une partition validant $\mathbf{P}$. L'interprétation de cet ensemble à l'aide des paires filtre-idéal permet en outre de révéler

les propriétés intrinsèques des opérateurs du treillis des partitions et d'étendre l'éventail des raisonnements applicables dans cette structure.

De plus, les opérateurs $l(.)$ et $r(.)$, ainsi définis pour les partitions, assurent la conformité de l'opérateur de $(\vee)$ vis-à-vis de sa sémantique. Or, il est possible de modifier leur sémantique selon le contexte applicatif dans lequel sont engendrées les partitions (*c.f.* par exemple l'étude [48]). Enfin, il serait intéressant d'étudier les propriétés axiomatiques du consensus dans le contexte général des treillis à l'aide de ces outils algébriques afin d'améliorer la compréhension du processus de construction d'un consensus.

6

La partition dans les SGBD

Ce chapitre est une version longue de l'article paru dans [45]. Il présente nos résultats sur l'opérationnalisation de l'algèbre des partitions à l'aide d'un système de gestion de base de données.

Comme nous l'avons vu dans les deux chapitres précédents, les partitions sont une manière naturelle de conceptualiser l'organisation des données. Cependant, nous manquons à ce jour de fonctionnalités propres à faciliter leur gestion au sein d'un environnement de développement. Nous proposons ici une avancée dans la compréhension de ce problème ainsi que les éléments pour mener à bien sa résolution.

En particulier, nous fondons notre problématique sur l'efficacité du traitement, de l'évaluation et de l'optimisation des requêtes sur des partitions dans le cadre des bases de données relationnelles. La production de plans d'exécutions efficaces demeure une gageure tant le modèle de représentation des données contraint la définition algébrique des opérateurs puis la complexité de leur mise en œuvre en ce qui concerne la complexité spatiale et temporelle.

Nous allons en premier démontrer qu'il n'existe pas d'encodage ou de représentation résolument triviale facilitant la gestion d'un ensemble de partitions. Nous allons également motiver l'encodage relationnel et montrer qu'il n'est pas possible d'exprimer des opérateurs et ceux du treillis des partitions comme requêtes de l'algèbre relationnelle. Par ailleurs, nous explorons des fonctions du langage SQL au-delà des requêtes du premier ordre afin de construire des requêtes optimisées pour le traitement de certains opérateurs. Nous mettons également en avant de nombreux cas où les requêtes du premier ordre ne sont jamais en mesure de fournir un cadre adéquat pour leur traitement, ce que nos résultats expérimentaux tendent à renforcer malgré l'usage de l'optimiseur

de requêtes SQL.

6.1 Introduction

Malgré leur omniprésence dans les représentations conceptuelles dans de nombreux problèmes, les partitions ne sont pas encore considérées comme une structure de données à part entière et par conséquent, elles ne peuvent être manipulées telles quelles en dépit de leurs nombreux usages. Cela devient plus critique encore alors qu'augmente considérablement le nombre de données à partitionner. Étant donné que les données utilisées dans un contexte scientifique – mais pas uniquement – tendent aujourd'hui à être intégrées au sein de dépôts collaboratifs, on peut également penser qu'à terme, les résultats d'analyses exploratoires par des méthodes de classification non supervisée seront également partagés et en nécessiteront ainsi une représentation unifiée.

Suivant cette perspective, nous imaginons que la mise à disposition de ces données dans le but d'élaborer des comparaisons ou des combinaisons, entre des partitions, profiterait de

1. l'indépendance entre la représentation logique et physique des partitions d'ensembles par l'emploi d'une couche d'abstraction ;
2. la disponibilité d'un langage approprié d'interrogation de partitions ;
3. un moteur d'évaluation de requêtes adapté aux partitions ;

Par la suite, nous discutons le problème d'interroger des partitions de manière générale, c.-à-d. indépendamment de la nature de l'univers ou d'autres facteurs caractérisant ces constituants. En d'autres mots, étant donné un ensemble d'objets, la seule connaissance disponible permet de déterminer si deux objets $x, y \in \Omega$ sont liés l'un à l'autre au sein d'un même groupe ou cluster $(x \in [y])$. Aucune hypothèse n'est établie sur les valeurs de x et y ou la satisfaction de propriétés communes. Ce cadre, le plus général qui soit, rend possible la combinaison de multiples partitions sur un même univers, mais figurant des points de vue différents sans requérir un compromis ou une astuce pour fusionner les représentations. Par exemple, des objets stellaires peuvent être partitionnés selon leurs propriétés chimique, leur rendu visuel, leurs propriétés acoustiques, *etc*.

Les cas présentés dans le cadre de la modélisation de cubes OLAP, puis la modélisation d'opérateurs visant à réaliser un consensus parmi un ensemble de partitions, illustrent selon nous les besoins pour un système de gestion de données tenant compte des partitions d'ensembles.

Notre ligne directrice consiste à établir une représentation des partitions à l'aide du modèle relationnel de Codd [35]. Ce modèle est en effet déjà éprouvé et fournit des systèmes fiables avec des capacités d'interrogation sophistiquées, des politiques d'optimisations de requêtes et un cadre théorique fondé sur des bases solides.

En tant que contribution principale, nous allons démontrer que la gestion des partitions fonctionne raisonnablement lorsque celles-ci sont représentées par un encodage relationnel, qui semble ainsi le choix le plus approprié.

En vue de démontrer cela, nous allons suivre les étapes suivantes :

- Discussion sur la conception d'un schéma d'encodage relationnel pour les partitions, en se basant sur la relation d'appartenance ;
- Définition des requêtes relationnelles pour les principaux opérateurs sur les partitions d'ensembles par comparaison avec le schéma d'encodage retenu ;
- Optimisation des opérateurs et leur mise en œuvre par des requêtes relationnelles ;
- Évaluation des performances des requêtes SQL sur un ensemble de partitions utilisant un générateur réaliste ;
- Mise en œuvre d'un prototype où les partitions résident en mémoire centrale et des expérimentations sont effectuées avec les opérateurs.

Dans la prochaine section, nous revenons brièvement sur quelques travaux relatifs à la représentation et à la manipulation de sous-ensembles d'ensemble, et apparaissant dans des contextes variés.

6.2 Travaux liés

Nous revenons dans cette section sur quelques travaux antérieurs visant à fournir un cadre ou une méthodologie permettant la gestion d'ensembles au sein de SQL. Nous y favorisons en particulier les approches conceptuelles et théoriques qui font référence à la construction et au maintien d'ensembles dans des contextes applicatifs variés. De fait, la littérature est généralement avare dans ce domaine, les références habituelles sur la construction de partition d'un ensemble utilisent une modélisation basée sur une structure *Union-Find* dont nous ferons justement usage à la fin de ce chapitre.

Néanmoins, de nombreux efforts ont déjà été effectués dans le but de gérer des ensembles par l'emploi de l'opérateur $(\subseteq)$ de comparaison d'ensembles et ses dérivés qui forment la relation canonique depuis laquelle les ensembles peuvent être mis en relation. Cette approche implique par ailleurs la conception de requêtes imbriquées multiples où chaque opérateur ensembliste nécessite d'être traduit en logique du premier ordre restreinte aux quantificateurs existentiels [64]. Cependant, l'évaluation de requêtes n'a pas été étudiée et ne couvre pas le cas des partitions dont les parties sont mutuellement disjointes.

Dans [111], les auteurs définissent le problème de la longueur de description minimale permettant de représenter une collection de sous-ensembles $\subset \mathcal{P}(\mathbb{U})$ à l'aide d'un

ensemble de symboles – les attributs d'un schéma relationnel – se trouvant en correspondance extensionnelle avec chaque ensemble particulier dans $\mathcal{X}$. L'application visée étant la construction d'expressions intensionnelles sur ces symboles minimisant le coût par rapport à une requête utilisateur équivalente dans le cadre d'un moteur d'optimisation de requêtes.

Une autre approche intéressante est celle de l'algèbre de partitions de relations, proposée dans [52]. Dans le domaine d'application du génie logiciel, cet outillage algébrique fournit un cadre fondamental dans le développement modulaire par l'emploi de l'opérateur binaire de *lifting*. Celui-ci est défini par une relation binaire sur les classes d'équivalence associées à une partition, c.-à-d. toutes les fonctions incluses dans le même module. On note cependant que ce travail met l'accent sur la couche relationnelle au-dessus de l'algèbre des partitions plutôt que la mise en œuvre de l'algèbre elle-même. Par exemple, la contribution principale concerne la dualité exprimée entre $X \times X$ et $X/\theta \times X/\theta$ qui met en relation les modules partageant les mêmes fonctionnalités et les modules eux-mêmes.

De plus, de nombreux travaux tels que [6, 26, 77] ont été menés dans la perspective de produire une correspondance entre des schémas relationnels et des schémas XML et objets, et suivant des orientations différentes. Bien que l'objectif soit semblable – ils essaient d'établir une correspondance entre des structures complexes et des relations –, aucune ne prête l'attention aux partitions d'ensembles.

Une orientation importante est le développement régulier du *groupjoin* depuis vingt ans [5, 93, 134]. Bien que la fusion d'une requête d'agrégation et de jointure relationnelle permette de calculer rapidement et efficacement le résultat des requêtes apparentées, indépendamment de son implémentation concrète, ces travaux ne possèdent pas d'extension permettant le calcul de la borne supérieure de deux partitions (*c.f.* la section dédiée).

Finalement, les partitions d'ensembles peuvent être vues comme des relations annotées [76]. L'annotation est alors essentiellement le représentant d'une classe d'équivalence, la relation principale identifiant les autres objets. Dans un tel contexte, les efforts ont été concentrés sur la définition de règles algébriques permettant la propagation d'annotations avec les opérations relationnelles sur les tables principales. Au contraire, l'encodage des partitions que nous proposons n'est pas conçu dans le but d'assurer la compatibilité avec des requêtes de l'algèbre relationnelle, mais plutôt d'apporter sa propre sémantique à travers le schéma d'encodage et les opérateurs de treillis associés.

Nous allons dans ce qui suit réaliser l'étude de l'encodage relationnel des partitions et des opérateurs que nous souhaitons mettre en œuvre.

6.3 Modèle de données

Nous motivons ici la représentation relationnelle des données. Celle-ci dépend principalement des propriétés combinatoires attachées aux partitions d'ensembles. Ces dernières comportent deux niveaux d'imbrication : une partition $P = \{a_1, a_2, \ldots, a_n\}$ est effectivement un ensemble d'ensembles (les a_i's sont des éléments de $\mathcal{P}(\Omega)$) d'objets tel que la relation induite soit exhaustive $\Omega = \bigcup_P a_i$ et les groupes mutuellement disjoints avec $a_i \cap a_j = \varnothing$ si $i \neq j$. Les notations utilisées pour décrire les partitions sont celles de la section idoine (treillis).

Une nouvelle fois, on utilise l'ensemble des entiers naturels $\mathbb{N}$ pour figurer l'univers des objets.

En comparaison des opérateurs naturels du treillis, et puisque chaque partition est également un ensemble d'ensembles, nous souhaitons pouvoir être en mesure d'appliquer des opérations ensemblistes sur les classes. En effet, les partitions peuvent être vues comme construites par la réunion de sous-ensembles de $\mathcal{P}(\Omega)$. Par conséquent, chaque partition est une famille de sous-ensembles disjoints pour lesquels nous pouvons appliquer les opérateurs suivants :

$$P \cap Q \triangleq \{a \mid a \in P \wedge a \in Q\}$$
$$P - Q \triangleq \{a \mid a \in P \wedge a \notin Q\}$$

L'opérateur d'intersection équivaut à calculer soit $P - (P - Q)$, soit $Q - (Q - P)$. $-$ et $\cap$ sont tout deux des opérateurs bien définis dès lors qu'ils induisent un nouvel univers $\mathcal{S} \subseteq \Omega$. En effet, seul un sous-ensemble des classes de P composera l'ensemble renvoyé de $P \cap Q$ ainsi que $P - Q$.

6.3.1 Représentation extensionnelle

Dans un premier temps, nous qualifions la représentation ensembliste d'une partition, d'intensionnelle. Par opposition, nous choisissons l'épithète extensionnelle pour désigner sa représentation équivalente, c.-à-d. conservant ses propriétés et donc la même interprétation. Nous allons nous intéresser à la conception d'un schéma bijectif ε qui transporte la structure entre chaque représentation.

La conception d'une représentation relationnelle d'une partition d'ensemble doit être réalisée suivant quelques prérequis. Selon un point de vue algébrique, une telle représentation devrait prendre uniquement en compte la connaissance nécessaire à la réalisation des affectations des objets de l'univers à leur classe respective. De plus, elle devrait permettre d'effectuer aisément le traitement de la requête suivante : à quelle classe appartient un objet donné d'une partition donnée ? Ceci consiste dans les faits à appliquer les opérateurs ensemblistes $\{\cap, -\}$ et par conséquent, l'exécution de tests booléens sur les classes.

Dans la même veine, les opérateurs latticiels $\{\wedge, \vee\}$ requièrent d'avoir à disposition la connaissance des classes et de leurs constituants. Ceci implique qu'une propriété devrait être préservée de manière à fusionner des objets à une classe sans avoir à explicitement réaliser des opérations d'emboîtement et d'aplatissement dans le but de changer de portée.

Décomposition d'une partition

Utilisant le théorème de représentation des treillis (*c.f.* 3.3.2), on réalise facilement que la structure de relations d'équivalence sur Ω est une extension valide pour n'importe quelle partition d'ensemble. Étant donnés deux objets $x, y \in \Omega$, on écrira $x \sim_P y$ si ils appartiennent à la même classe dans la partition P. On obtient alors la construction classique définie par :

$$\phi : \Pi_\Omega \to \mathcal{P}(\Omega \times \Omega) \tag{6.1}$$

$$P \mapsto \bigcup_{(a,a)\in P\times P} a \times a \tag{6.2}$$

et donnant $\sim_P \triangleq \phi(P)$.

La construction quotient nous permet dès lors de retrouver la partition originale en définissant son inverse par :

$$\phi^{-1} : \mathcal{P}(\Omega \times \Omega) \to \Pi_\Omega \tag{6.3}$$

$$\sim_P \mapsto P/\sim_P \tag{6.4}$$

puis de recouvrer l'ensemble des affectations originales des objets à leur classe. Comme nous l'avons vu précédemment, cette représentation est maximale en terme d'espace puisqu'en $O(n^2)$ – même en appliquant la réduction réflexive et symétrique. On voit aisément que ϕ transpose le calcul entre partitions sur les relations d'équivalence associées :

$$\phi(P \wedge Q) = \sim_P \cap \sim_Q$$
$$\phi(P \vee Q) = \langle \sim_P \cup \sim_Q \rangle^+$$

où $\langle . \rangle^+$ figure la fermeture transitive de $\sim_P \cup \sim_Q$, nécessaire à maintenir l'ensemble résultat comme étant fermé pour l'union.

Puisque chaque $x \in \sim_P$ est une paire d'éléments sur $\Omega \times \Omega$, il figure un atome indiquant s'ils sont liés dans une même classe (*c.f.* également les chapitres précédents sur l'usage de $\mathcal{P}(\Omega \times \Omega)$).

Tandis que les éléments sont distincts les uns des autres au sein d'une relation d'équivalence, il existe toujours des objets de l'univers qui sont partagés dans deux partitions différentes. En l'absence de l'équation de distributivité, la représentation ensembliste d'une partition par une collection d'atomes implique en règle générale que le calcul de l'opérateur $(\vee)$ nécessite l'ajout de nouveaux atomes symbolisant la fermeture transitive des classes.

S'affranchir de cette opération est alors uniquement possible si les deux partitions ne contiennent pas de classes se chevauchant. Par conséquent, la propriété suivante est vérifiée :

$$P \vee Q = P \vee R \implies Q \neq R$$

Cela signifie que pour toute paire de partitions P et Q, non réduites à un unique atome, il existe toujours plusieurs décompositions minimales de même cardinalité dont on ne peut déduire de manière déterministe une représentation cohérente comme étant la meilleure solution et préservant $P \vee Q$. Nous avons donc à construire en ne partant de rien, une représentation qui convient mieux aux lois de composition des partitions.

Le problème est donc similaire à la recherche d'un pseudo-complément :

$$R \triangleq Q - P = \bigwedge \{X \mid Q \leq P \vee X\}$$

qui n'est pas défini dans Π_Ω et ne permettant pas d'écrire :

$$P \vee R = Q$$

où R déterminerait la plus petite partition en ce qui concerne le nombre d'atomes pour atteindre Q et telle que :

$$P \leq P \vee R \wedge Q \leq P \vee R$$

Transcrire les opérations du treillis des partitions $\mathsf{Op} = \{\wedge, \vee\}$ à l'aide des opérateurs ensemblistes $\mathsf{EnsOp} = \{\cap, \cup\}$ se formalise alors par l'identité suivante :

$$\phi(P \ \mathsf{Op} \ Q) = \phi(P) \ \mathsf{EnsOp} \ \phi(Q) \tag{6.5}$$

$$\iff \phi \circ \ \mathsf{Op} \ (P, Q) = \ \mathsf{EnsOp} \ \circ (\phi(P), \phi(Q)) \tag{6.6}$$

En revanche, lorsque $\mathsf{Op} = \{\cap, -\}$, l'application de ces opérateurs ne peut être réalisée depuis les relations d'équivalences $\phi(P)$ et $\phi(Q)$. En effet, les appartenances explicites des objets à leur classe ne sont plus disponibles sous cette représentation et cela nécessite donc de les recalculer à la volée.

Un problème crucial est d'éviter ce genre de calcul lorsqu'on souhaite retrouver les appartenances explicites, puisque cela nécessite de calculer :

$$\phi(P \ \ \mathsf{Op} \ \ Q) = \phi(\phi^{-1}(\sim_P) \ \ \mathsf{Op} \ \ \phi^{-1}(\sim_Q)) \tag{6.7}$$

$$\iff \phi \circ \mathsf{Op}(P, Q) = \phi \circ \mathsf{Op}(\phi^{-1}(\sim_P), \phi^{-1}(\sim_Q)) \tag{6.8}$$

et réduit à néant l'intérêt d'une telle représentation.

Représentation arborescente

Heureusement, puisque nous avons affaire à des relations strictes – un objet appartient soit tout à fait soit pas du tout à une classe –, où chaque objet est indiscernable des autres au sein de la même classe, la propriété de transitivité peut être relâchée sous

cette hypothèse en une propriété d'accessibilité. Étant donnée une relation d'équivalence $\sim_P \subseteq \Omega \times \Omega$ représentant une partition, il n'est donc pas nécessaire de maintenir l'ensemble des atomes matérialisant le lien entre chaque constituant de la classe, mais seulement un sous-ensemble $E \subseteq \Omega \times \Omega$ tel que :

$$\langle E \rangle^+ = \sim_P$$

En employant le vocabulaire de la théorie des graphes, nous allons considérer comme cohérent le fait que les sommets x, y, z sont équivalents selon une partition donnée s'il existe un chemin dirigé veillant à ce que la classe forme une composante fortement connexe plutôt qu'un sous-graphe complet.

Une telle structure est alors optimale en espace, c.-à-d. $O(n)$, si chaque classe est mise en œuvre par un arbre couvrant. Cela implique que chaque objet au sein d'une classe possède au moins deux arcs (à l'exception des feuilles) et ainsi signifie que n'importe quel objet peut être atteint depuis un autre, au sein de la même classe.

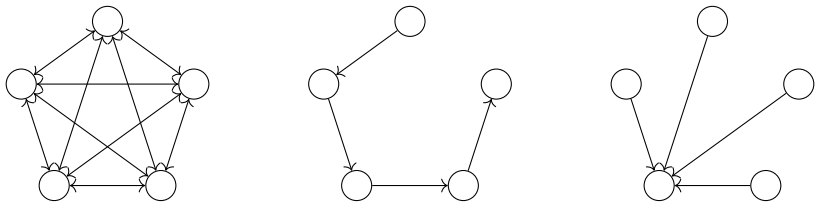


FIGURE 6.1 : À gauche, le graphe d'une relation d'équivalence, au milieu, celui d'un circuit brisé et à droite, un graphe étoile

Deux structures arborescentes ressortent particulièrement comme le montre la figure (6.1) :

- un circuit brisé, c.-à-d. l'ensemble des objets est regroupé le long d'un chemin orienté, l'un d'entre eux est choisi pour figurer la racine ;
- un arbre à un niveau, c.-à-d. un graphe étoile pour lequel un objet arbitraire est choisi parmi la classe pour être la racine.

Le cas du chemin orienté n'est pas pertinent et devrait être rejeté sachant qu'une recherche est en $O(n)$ dans le pire cas tandis que le cas de l'étoile garantit un accès en temps constant. De plus, une telle structure peut être aisément balayée à travers un encodage relationnel puisqu'elle est maximalement aplatie.

En effet, le choix d'un ensemble de représentants pour une partition P définie sur Ω doit être effectif pour l'application des opérateurs et les transformations ensemblistes appliquées sur les classes de chaque opérande. Regardant l'univers comme un ensemble d'alternatives, on définit la fonction de choix $c : \mathcal{P}(\Omega) \to \mathcal{J}(\mathcal{P}(\Omega))$ qui choisit un représentant parmi un ensemble candidat $\mathcal{J}(\mathcal{P}(\Omega))$, identifiant une classe $X \in \mathcal{P}(\Omega)$. Étant donné un ensemble $X \in \mathcal{P}(\Omega)$, on a les identités suivantes :

$$c(X) \in \mathcal{J}(\mathcal{P}(\Omega)) \tag{6.9}$$

$$c(X) \subseteq X \tag{6.10}$$

La première identité est évidente et exprime le fait que chaque représentant est unique pour chaque classe et est identifié à l'aide d'un atome, en particulier $\mathcal{J}(\mathcal{P}(\Omega)) \cong \Omega$: pour tout $x \in \Omega, \{x\} \in \mathcal{J}(\mathcal{P}(\Omega))$. La seconde est la conséquence de la première où le représentant de la classe est nécessairement un de ses constituants.

Puisque le calcul de la partition figurant le résultat d'une opération algébrique peut être construit de manière itérative, cela implique la construction d'une hiérarchie de partition dont le sommet figure alors le point fixe commun aux opérandes. Étant donné $\mathcal{X} = \{X_1, X_2, \ldots, X_n\}$, un sous-ensemble des classes d'une partition P dont on doit réaliser l'union, il existe alors un nombre exponentiel de séquences ou d'ordres sur $\mathcal{X}$ pour réaliser $\bigcup X_i$.

Or, le choix du représentant de chaque X_i doit être indépendant de l'ordre induit sur $\mathcal{X}$ pour construire le résultat. La condition d'indépendance de Plott[86] (*path independence*) formalise cette propriété par l'identité suivante :

$$c(\bigcup_i X_i) = c(\bigcup_i c(X_i))$$

Les fonctions de choix vérifiant cette propriété garantissent que l'élément retourné par $c(\bigcup_i X_i)$ sera le même que celui calculé successivement pour chaque paire d'ensembles $(X_i, X_j) \in \mathcal{X}^2$. Un opérateur vérifiant cette condition permet donc de choisir un ensemble de représentants pour chaque classe d'une partition (une *ancre* désigne un représentant par la suite). Nous présenterons les propriétés de cette structure dans la section (6.6).

6.3.2 Encodage relationnel

Ainsi que cela a été établi précédemment, le modèle relationnel permet nativement l'encodage des partitions et fournit ainsi un schéma d'encodage direct pour une représentation à base d'arbres. Nous faisons référence à l'encodage relationnel lorsque nous discutons du schéma d'encodage de la relation d'appartenance des objets au sein des classes. En particulier, nous discuterons uniquement du schéma d'encodage introduit dans la dernière section et élaboré sur la base d'une représentation arborescente.

Encodage de la relation d'appartenance d'une partition

Le schéma d'encodage nécessite un schéma pourvu de deux colonnes, un pour l'identifiant d'objet, le second pour l'identifiant de classe. Ces derniers peuvent être engendrés par le système, cependant, nous fournissons ceux-ci parmi le même univers que celui des objets. Cela n'entrave pas les capacités d'encodage et n'a pas d'impact sur la taille de la représentation. Cela permet de plus de formuler les requêtes associées aux opérateurs de manière efficace (*c.f.* Section 6.4).

Définition 6.3.1 (Relation d'appartenance)**.** *Soit une partition P et sa relation d'équivalence associée $\sim_P$; soit le schéma $S(\mathsf{elt} : \Omega, \mathsf{classe} : \Omega)$ où les colonnes* elt *et* classe *sont*

M	
elt	classe
1	1
2	1
3	1
4	4
5	4
6	6

(a)

N	
elt	classe
1	1
2	1
3	1
4	4
5	5
6	5

(b)

FIGURE 6.2 : Vue relationnelle de $P = 123|45|6$ et $Q = 123|4|56$

définies dans Ω. Le schéma d'encodage relationnel ε des partitions d'ensemble est défini par :

$$\begin{array}{rccl} \varepsilon: & \Pi_\Omega & \mapsto & S(\Omega,\Omega) \\ & P & \to & \mathbf{I}(S) \triangleq \{(x,y) \mid x \in [y]\} \end{array}$$

où $\mathbf{I}(S)$ désigne une instance de la relation S.

Dans la définition du dessus, nous imposons que chaque classe d'équivalence $[y]$ de $\sim_P$ possède une *ancre* y, c.-à-d. un objet saillant qui identifie la classe. Avec les propriétés algébriques établies dans la section (6.3.1), nous décidons arbitrairement de fixer l'ancre à la plus petite valeur parmi la classe, présumant que $[y]$ possède une unique borne inférieure tandis que la condition d'indépendance des chemins implique que ε doit être définie comme une fonction extrême, et par la suite, y peut se voir affecter soit la valeur $\min$ soit la valeur $\max$ uniquement.

C'est en particulier nécessaire pour assurer que $\varepsilon(P)$ est unique quelle que soit P. Il s'ensuit que les objets, à l'exception de l'ancre, forment une fratrie au sein de l'arbre représentant la classe. Donc les objets à l'exclusion de la racine ne sont pas ordonnés.

Les tables M et N de la figure (6.2) sont des instances du schéma S et illustrent la représentation relationnelle des partitions $P = 123|45|6$ et $Q = 123|4|56$. Les identifiants de classes sont 1,4,6 pour P et 1,4,5 pour Q. Ce sont les valeurs minimales de leurs classes respectives. Chaque classe, telle que 123 dans P, est encodée par un ensemble d'arêtes, p.ex. $\{(11),(12),(13)\}$, de sorte que cela induise un arbre à un niveau – un graphe étoile – où la racine est l'ancre (1).

6.3.3 Discussion

La contribution présentée dans ce chapitre porte sur la possibilité d'adopter un encodage relationnel. Celui-ci ouvre donc la voie à une implémentation concrète suivant un grand nombre de SGBDs ou de bases de données NoSQL qui permettent tous une représentation tabulaire des partitions.

Le modèle relationnel s'est imposé depuis les années 70 comme le modèle de représentation de données structurées. Il est cependant apparu que ce modèle comportait des limites (calcul de fermeture, gestion des objets complexes) par trop contraignantes pour de plus en plus d'applications. De nouveaux modèles ont donc été proposés pour y remédier. Parmi eux, les objets complexes, les arbres et les graphes sont trop expressifs pour calculer efficacement des requêtes sur des partitions d'ensembles tandis que le modèle clé/valeur (put/set) est par trop restrictif.

Certaines recherches en cours ont été réalisées dans le but d'enrichir le développement de la modélisation de structures de données complexes par le biais d'encodage relationnel. Bien que la difficulté du modèle relationnel pour l'encodage de données complexes ait déjà été soulignée, les moteurs de requêtes relationnelles appuyés par l'algèbre relationnelle sont sans le moindre doute les plus étudiés et ont obtenu une importante popularité de la part des chercheurs et des professionnels travaillant dans le domaine des systèmes de bases de données. On peut citer par exemple le cas des données multidimensionnelles suivant le modèle R-OLAP (*c.f.* Chapitre 3) ou les imbrications d'ensembles permettant le stockage d'arbre XML.

Le modèle relationnel imbriqué et son algèbre sont en principe des alternatives évidentes pour traiter des partitions d'ensembles génériques (ensembles d'ensembles). En revanche, la mise en œuvre des opérateurs associés nécessiterait un accroissement du coût de traitement lié à des opérations superflues d'emboîtement et d'aplatissement pendant l'exécution de la requête, sans apporter de contrepartie acceptable à l'exception de la pertinence du modèle imbriqué pour représenter nativement des partitions.

Les partitions peuvent également être représentées par des hypergraphes déconnectés où les hyperarêtes sont les classes. De récents développements dans les bases de données graphes ont mis l'accent sur les structures de données comme les graphes RDF et permettent la recherche de sous-graphes ainsi que des tests de connectivités par le biais de requêtes de chemins.

Il est donc délicat de réduire la liste des différentes alternatives acceptables en éliminant celles n'étant pas conformes à nos attentes. En effet, la définition du schéma d'encodage que nous adoptons ici ne tirera pas nécessairement avantage des différentes fonctionnalités offertes par le modèle ACID ou de l'optimiseur de requêtes d'un SGBD. En conséquence de quoi, il semble préférable d'adopter plutôt une base de données clé/-valeur, au moins comme un support de persistance. En contrepartie, il est vraisemblable qu'il soit difficile d'adapter des requêtes complexes (jointures, sous-requêtes imbriquées et récursives) efficacement. Comme nous allons le voir dans l'implémentation SQL des opérateurs de la prochaine section, la mise en œuvre des opérateurs est plutôt laborieuse et il semble peu probable de pouvoir implémenter efficacement nos opérateurs à travers une base de données de ce type.

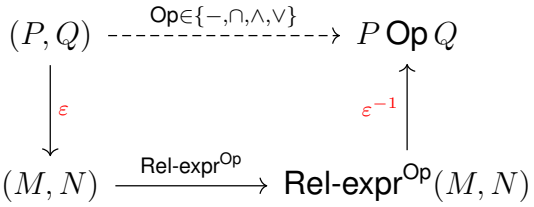


FIGURE 6.3 : Diagramme de l'encodage relationnel

6.4 Réalisation des opérateurs

Dans cette section, nous allons nous focaliser sur le moyen de traduire les opérateurs $\{-, \cap, \wedge, \vee\}$ par le biais de requêtes relationnelles et utilisant notre schéma d'encodage ε.

On part du principe que deux partitions P et Q sont respectivement encodées par les relations M et N par la suite. Nous considérons également que les partitions P et Q sont définies sur le même univers Ω.

La figure (6.3) donne une vue d'ensemble du point que nous traitons dans cette section. L'idée principale est de fournir pour chaque opérateur Op, une requête $\mathsf{Rel\text{-}expr}^{\mathsf{Op}}$ formulée dans un langage relationnel quelconque, telle que l'expression $P\,\mathsf{Op}\,Q$ soit donnée par :

$$P\,\mathsf{Op}\,Q = \varepsilon^{-1}(\mathsf{Rel\text{-}expr}^{\mathsf{Op}}(\varepsilon(P), \varepsilon(Q)))$$

Nous suivons en cela le schéma classique d'encodage des requêtes XPath en SQL.

Plus précisément, trois difficultés doivent être levées afin de traiter l'équivalent relationnel des opérateurs agissant sur les partitions d'ensembles : (i) la *comparaison* des classes, (ii) l'*intersection* des classes et (iii) l'*identification* des classes.

Initialement, l'application des opérateurs ensemblistes $\{-, \cap\}$ sur les partitions nécessite de tester l'égalité entre les classes de M et celles de N, deux à deux. Nous devons également décider si une classe de M chevauche une classe de N, en particulier lorsque l'on souhaite réaliser l'opération $(\vee)$ qui fusionne les classes de deux partitions qui se chevauchent mutuellement. Ces opérations sont effectuées par des comparaisons de classes soit l'application de prédicats sur celles-ci.

Puis, le traitement de l'opérateur $(\wedge)$ implique de calculer l'ensemble de toutes les intersections entre chaque paire de classes de M et N. Le mécanisme de base nécessite de calculer les relations d'équivalence $\phi(P)$ et $\phi(Q)$ depuis M et N à la volée. Cette opération est facilement réalisée par l'expression $\pi_{1,3}\sigma_{2=4}(M \times M)$ bien qu'elle occasionne une sévères chute des performances (*c.f.* section 6.5).

Le dernier point, mais non des moindres, part du constat que tous les opérateurs latticiels nécessite l'affectation d'une ancre (identifiant de classe) pour chaque nouvelle classe créée durant le processus. Par conséquent, le traitement doit corriger récursivement les identifiants de classe par leur valeur minimum.

6.4.1 Opérateur de différence

Nous définissons l'opérateur de *différence* à l'aide du calcul relationnel de domaine (*Domain Relational Calculus*) par l'expression suivante :

$$\varepsilon(P-Q) \triangleq \{(x,y):\ M(x,y) \wedge \forall z.\exists t.\neg\,(M(t,y) \leftrightarrow N(t,z))\}$$

où nous construisons les paires éléments-classes de sorte que chacune d'entre elles soit dans M et que nous ne puissions pas trouver de classe z dans N qui soit égale à la classe y dans M.

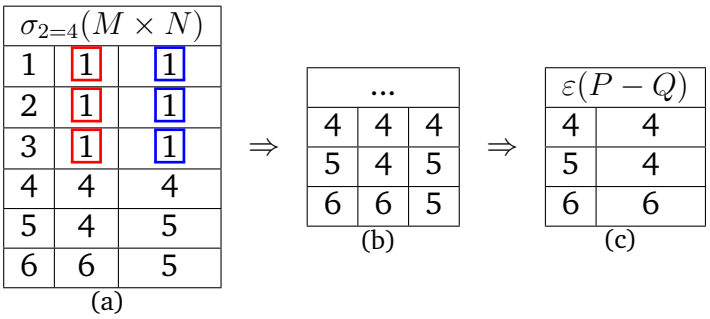


FIGURE 6.4 : $\varepsilon(P-Q)$: la première classe de M correspond à la première de N

Le principe utilisé par l'opérateur de différence est illustré par la figure (6.4) et emploie les partitions P et Q de la figure (6.2).

Nous proposons maintenant l'expression algébrique suivante : :

$$\begin{aligned}\varepsilon(P-Q) \triangleq & M - \pi_{1,2}(\sigma_{2=3}(M \times ((\mathbf{adom} \times \mathbf{adom}) - \\ & \pi_{2,4}(\sigma_{1=3}(((\mathbf{adom} \times \mathbf{adom}) - N) \times M) \cup \\ & \sigma_{1=3}(((\mathbf{adom} \times \mathbf{adom}) - M) \times N)))))\end{aligned}$$

où $\mathbf{adom} = \text{adom}(M) = \text{adom}(N) = \pi_1(M) = \pi_1(N)$ étant donné que l'univers des deux partitions P and Q est le même. L'évaluation naïve de $\varepsilon(P-Q)$ va donc nécessiter le calcul de trois produits cartésiens, autant d'équi-jointures et différences ensemblistes et une opération d'union appliquée sur des relations de taille $O(|\mathbf{adom}|^2)$.

Dans la même veine, puisque l'équivalence $A \cap B \equiv A - (A - B)$ tient, alors l'opération de différence permet de définir de manière appropriée l'opérateur d'*intersection*. Nous sommes, de plus, capable de fournir une définition autonome pour $(\cap)$ à l'aide de la requête suivante :

$$\varepsilon(P \cap Q) \triangleq \{(x,y):\ M(x,y) \wedge \exists z.\forall t.\,(M(t,y) \leftrightarrow N(t,z))\}$$

Par ailleurs, on observe que l'expression ci-dessus possède la même sémantique lorsque les paramètres M et N sont permutés. La seule différence réside dans le fait que les identifiants de classes de l'intersection proviennent indistinctement soit de M, soit de N. En outre, les classes figurant dans le résultat de $P \cap Q$ ont nécessairement le même identifiant dans l'une ou l'autre partition.

6.4.2 Opérateur de borne inférieure

Le calcul de la borne inférieure $P \wedge Q$ se traduit en l'expression suivante :

$$\begin{aligned}\varepsilon(P \wedge Q) \triangleq \{(x,y) : \exists z.(M(x,z) \wedge M(y,z) \wedge \\ \exists t.(N(x,t) \wedge N(y,t) \wedge \\ \forall u.((M(u,z) \wedge N(u,t)) \rightarrow u \geq y)))\}\end{aligned}$$

La formule ci-dessus combine le problème de l'intersection des classes avec le problème de leur identification. En effet, affecter une valeur particulière y jouant le rôle d'ancre des différentes paires (z,t) est opérationnellement similaire à l'affectation d'une ancre à un ensemble d'objets équivalents entre eux. Concrètement, les paires (z,t) définissent à unicité près chaque nouvelle classe dans la partition résultante de l'opération. Cela nécessite alors de recalculer l'ensemble des classes d'équivalence depuis le schéma d'encodage.

Par exemple, depuis les partitions de la figure (6.2), on peut observer sur la figure (6.5) que chaque paire, constituée de classes de M et de classes de N, identifie une classe résultante de l'opération $\varepsilon(P \wedge Q)$. L'étape suivante pour réaliser le résultat est d'affecter les ancres aux classes. Cela est illustré sur la figure (6.6) par l'auto-jointure de $\mathrm{JTab} = \sigma_{2=4}(M \times N)$.

$\sigma_{2=4}(M \times N)$		
1	1	1
2	1	1
3	1	1
4	4	4
5	4	5
6	6	5

(a)

FIGURE 6.5 : $\varepsilon(P \wedge Q)$: les paires d'identifiants de classes de M et N

6.4.3 Opérateur de borne supérieure

Le calcul de la borne supérieure $P \vee Q$ ne peut être exprimé dans l'algèbre relationnelle puisqu'une opération de fermeture transitive est nécessaire pour mener à bien le traitement [42].

En effet, chaque union est propagée vers les classes à chaque fois que deux classes de P et Q se chevauchent, jusqu'à ce qu'un point fixe soit atteint. Puisqu'il n'est pas possible de planifier *a priori* le nombre d'itérations nécessaires pour propager les associations manquantes, alors aucune expression de l'algèbre relationnelle ne pourra calculer $\varepsilon(P \vee Q)$.

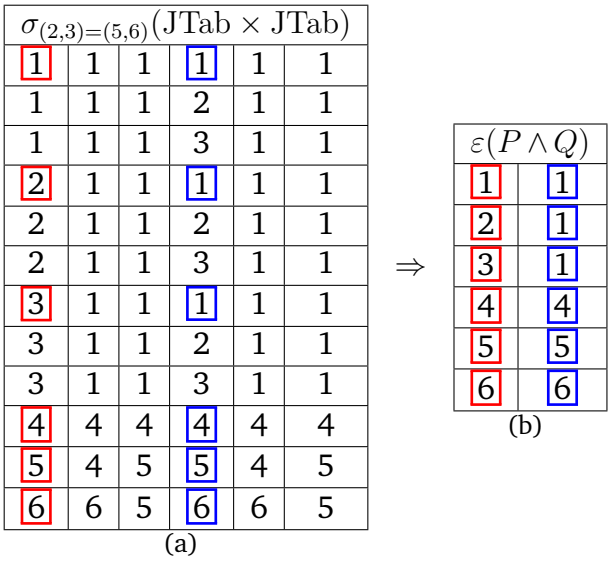


FIGURE 6.6 : $\varepsilon(P \wedge Q)$: les objets et leur ancre, avec $\mathrm{JTab} = \sigma_{2=4}(M \times N)$

Cependant, Datalog est connu comme contenant strictement l'algèbre relationnelle et inclus également des fonctionnalités de récursion, aptes à la réalisation de l'opération. Nous allons donc concevoir un programme Datalog pour mettre en œuvre $\varepsilon(P \vee Q)$.

Essentiellement, deux étapes distinctes sont nécessaires pour mener à bien l'opération :

1. nous devons construire les connexions entre les identifiants de classes au sein d'une même opérande,
2. puis nous devons filtrer le résultat de sorte que chaque ensemble d'identifiants des classes d'équivalence possède la même ancre.

Suivant cette perspective, réaliser l'opération peut être vu comme l'élaboration d'une relation d'équivalence s'appliquant sur les classes elles-mêmes. Puis, de ces dernières, le mécanisme de calcul des ancres est réalisé par une étape de réduction.

La première étape ne peut donc pas être exprimée en algèbre relationnelle puisqu'elle nécessite de modéliser une requête d'accessibilité dans le graphe d'une relation binaire ; tandis que la seconde étape admet, elle, une formulation dans ce langage. Le programme Datalog réalisant la première étape est défini ainsi :

$$\begin{aligned} r_1 &: \quad \theta(y, y) \leftarrow M(x, y) \\ r_2 &: \quad \theta(x, y) \leftarrow \theta(x, z), M(t, z), N(t, u), N(v, u), M(v, y) \end{aligned}$$

Chaque règle Datalog est sûre et indépendante du domaine et l'absence de négation fait que le programme est trivialement stratifié. La seconde règle apporte la récursivité où θ apparaît en tête et dans le corps de la règle. À la fin, θ est une relation d'équivalence sur les classes de M.

La seconde étape du traitement permet de filtrer les tuples de θ puis d'établir une ancre pour chacune des classes retenues dans θ, et enfin d'affecter les ancres aux objets. La requête associée est ainsi définie par l'expression suivante :

$$\varepsilon(P \vee Q) \triangleq \{(x,y) : \exists z.M(x,z) \wedge \theta(z,y) \wedge \\ \forall t.(\theta(z,t) \rightarrow t \geq y)\}$$

En guise de résumé, nous esquissons l'algorithme qui réalise $\varepsilon(P \vee Q)$, dans lequel nous mélangeons style procédural (figuré par la boucle) et calcul relationnel de domaine dans l'algorithme (6.4.1). Lors du traitement, θ^{ω} figure le point fixe qui est atteint en un nombre fini d'étapes puisque la séquence $(\theta^{(i)})_i$ est croissante et monotone ($\theta^{(i)} \subseteq \theta^{(i+1)}$). Enfin, $|\mathbf{adom} \times \mathbf{adom}|$ borne supérieurement la cardinalité de θ^{ω}.

Algorithme 6.4.1 Calcul de la borne supérieure $\varepsilon(P \vee Q)$

$\theta^{(0)} \leftarrow \{(y,y) : \exists x.M(x,y)\}$ ▷ Étape d'initialisation
répéter
$\theta^{(i+1)} \leftarrow \theta^{(i)} \cup \{(x,y) : \exists z.(\theta^{(i)}(x,z) \wedge \exists t.(M(t,z) \wedge \exists u.(N(t,u) \wedge \exists v.(N(v,u) \wedge M(v,y)))))\}$
$i++$
jusqu'à $\theta^{(i+1)} = \theta^{(i)}$
$\theta^{\omega} \leftarrow \theta^{(i)}$
retourner $\{(x,y) : \exists z.M(x,z) \wedge \theta^{\omega}(z,y) \wedge \forall t.(\theta^{\omega}(z,t) \rightarrow t \geq y)\}$

L'analyse ci-dessus justifie la finalité d'une mise en œuvre pratique de $\varepsilon(P \vee Q)$ dans un SGBD relationnel. Sachant que la norme ANSI/ISO SQL3 introduit la clause `WITH RECURSIVE` qui étend l'expressivité de SQL en lui permettant de simuler la récursivité de Datalog, nous sommes alors en mesure d'exprimer une unique requête SQL par une expression de table commune (*Common Table Expression*) pour réaliser l'opérateur de $(\vee)$ entre des partitions.

Cependant, cet algorithme est susceptible d'être contre-performant à cause de la collecte d'un nombre superflu de tuples dans θ. De plus, lors de la dernière étape récursive menant à $\theta^{(i+1)} = \theta^{(i)}$, la relation contient alors, dans le pire des cas, toutes les solutions candidates d'identifiants des blocs sur la totalité du domaine actif. C'est en particulier le cas lorsque toutes les classes doivent être fusionnées en une seule dans la partition résultante, ce qui mène à vérifier $|\pi_2(M)|^2$ tuples.

Utilisant pour nos expériences un générateur réaliste (*c.f.* notre protocole expérimental dans la section 6.5.2), le nombre de classes construites dans les partitions est de l'ordre de $\log^2(|\pi_1(M)|)$. On aura donc $\log^2(|\Omega|)$ tuples à vérifier. Finalement, pour tous les éléments du domaine actif dont l'identifiant de classe doit être réaffecté, on doit éliminer toutes les solutions *ex æquo* et choisir le minimum comme nouvel identifiant de classe. Au pire cas, cela nécessite $|\Omega| \times \log^2(|\Omega|)$ comparaisons ; le coût du calcul est alors prohibitif.

Par ailleurs, l'espoir d'améliorer ce coût par le biais d'optimisations appliquées à des requêtes récursives (p.ex. [102]) est infondé puisqu'elles conviennent uniquement aux

cas où le calcul de requêtes du genre de la fermeture transitive est espéré (telles que le calcul des chemins dans un graphe). Notre but n'est en aucun cas la fermeture transitive, elle est simplement un outil utilisé dans une étape transitoire.

La *réduction transitive* représente notre objectif final en réalisant les nouvelles affectations en fin de traitement. Dans la prochaine section, nous proposerons une méthode alternative basée sur la structure Union-Find, permettant la modélisation conjointe de la fermeture et réduction transitive à un faible niveau d'abstraction.

6.4.4 Optimisations suivant des fonctionnalités SQL

Afin d'améliorer les performances de l'évaluation des requêtes associées aux opérateurs sur les partitions, nous allons discuter de deux possibilités qui s'appliquent soit à tous les opérateurs, soit à un sous-ensemble.

L'idée simple retenue pour ces optimisations est d'utiliser les capacités d'indexations offertes par le SGBD relationnel. Cependant, puisque les partitions sont supposées être combinées les unes aux autres à l'aide d'expressions complexes, l'ensemble des optimisations applicables sera mis en place uniquement pendant l'exécution de la requête et aucune information auxiliaire ne devrait être considérée (hormis ce que révèle l'encodage en lui-même).

Filtrage précoce

La première technique d'optimisation traite des opérateurs ensemblistes $(-)$ et $(\cap)$. Elle consiste en une étape de préfiltrage au cours de laquelle les classes ne pouvant être incluses dans le résultat sont au préalable écartées du calcul.

En effet, lors de l'exécution de la requête pour les opérations ensemblistes $P \,\mathsf{Op}\, Q$, $\mathsf{Op} \in \{-, \cap\}$, chaque paire d'identifiants de classe (z, t) est vérifiée en balayant tous ses constituants afin de décider si les classes sont égales ou non. Bien que ce balayage intégral demeure nécessaire pour quelques paires candidates, il est possible d'éliminer certaines paires par l'emploi d'un prédicat élémentaire : $F(z) = F(t)$ avec $F \in \{\min, \max, \mathrm{count}\}$ sachant que ces fonctions d'agrégat sont mises en œuvre efficacement dans les SGBD relationnels.

Dans une moindre mesure, il serait pratique de définir une *signature* des classes à l'aide de telles fonctions, agissant comme une empreinte sur les objets de la classe à la manière d'un filtre de Bloom et vérifiant l'équation suivante :

$$sgn(X) \neq sgn(Y) \implies X \neq Y$$

où $X, Y \in \mathcal{P}(\Omega)$.

Par exemple, soient les partitions $P = 123|457|6|89$ et $Q = 123|467|58|9$; nous souhaiterions réaliser $P \cap Q$. Il y a *a priori* douze paires d'identifiants de classes à traiter. Si

on emploie les statistiques de cardinalités de classes, seules six paires sont candidates pour figurer des classes identiques aux deux opérandes :

$$\{(123, 123), (123, 467), (457, 123), (457, 467), (6, 9), (58, 89)\} \in P \times Q$$

Utilisant le minimum (soit l'identifiant de la classe), il demeure ensuite uniquement deux paires candidates :

$$\{(123, 123), (457, 467)\}$$

La valeur maximale ne jouera elle aucun rôle bénéfique pour rejeter les mauvaises paires. À la fin, l'étape de raffinement devrait permettre d'éliminer la seconde paire candidate afin de déduire que $P \cap Q = 123$.

Nous expérimenterons une version révisée des requêtes SQL qui tiendra compte du calcul rapide de ce genre d'agrégats $(\min, \max, \text{count})$ pour éliminer rapidement des paires ne pouvant faire partie du résultat.

Fonctions de fenêtrage

À l'instar de la récursivité, la norme ANSI/ISO SQL3 introduit également les fonctions de fenêtrage (*Window Functions*), soit une méthode raffinée d'agrégation qui est maintenant intégrée dans la plupart des SGBD relationnels.

L'idée est d'étendre les fonctionnalités permises par des requêtes d'agrégations par la clause `GROUP BY`, de sorte qu'il soit possible d'affecter chaque tuple à une valeur d'agrégat qui dépend d'un sous-ensemble variable de tuples.

Cette caractéristique peut être utile pour le calcul de la borne inférieure $P \wedge Q$ *via* notre schéma d'encodage. Dans notre cas, nous souhaiterions calculer la plus petite valeur de chaque classe et l'affecter à chaque tuple de chaque classe. Nous avons donc à construire une telle requête en partitionnant $M \bowtie_{M.1=N.1} N$ sur les paires qui correspondent à toutes les combinaisons d'intersections de classes.

Puisque ces paires identifient les classes dans le résultat, on peut affecter directement la valeur minimale de sous-ensembles de tuples.

6.5 Expérimentations

Dans cette section, nous effectuons le compte-rendu et la discussion de nos résultats expérimentaux. La principale conclusion à en tirer est la suivante : le moteur d'évaluation des requêtes SQL atteint des temps de traitement des requêtes incroyablement élevé pour n'importe quel jeu de données de taille modeste, à l'instar de ce que nous avions supposé dans les sections précédentes.

Les stratégies proposées dans la section 6.4.4 ont également été mises en œuvre afin d'appuyer nos conclusions sur l'inadéquation du calcul des partitions dans le cadre

offert par le langage SQL. Chacune d'entre elles améliore notoirement les performances globales des opérateurs associés grâce aux fonctionnalités avancées, proposées par le langage SQL. Néanmoins, *il n'existe pas de moyens permettant de contourner le calcul de la fermeture transitive, exigé par le calcul de la borne supérieure* $\varepsilon(P \vee Q)$ *dans ce cadre-là*.

Nous allons également alimenter nos résultats d'expériences avec ceux obtenus depuis la version des opérateurs (C++) que nous avons développés et qui surpassent leur pendant SQL tandis que se conformant avec les principes énoncés dans la section (6.4). Évidemment, une telle mise en œuvre ne peut véritablement simuler la modélisation de la gestion de la persistance des données ou le procédé de pagination quand les tables sont trop grosses pour résider en mémoire centrale.

Dans un tel cas, il nous est possible de mettre en œuvre un système de gestion de la mémoire distribuée qui prend en compte le cas des tables scindées, si le cas échoit, dans une unité de stockage indépendante et leur recherche à partir d'une table de hachage distribuée. Certains écosystèmes logiciels permettent, tel Hadoop, de combiner une base de données de type NoSQL comme support des données et utilisant le modèle MapReduce pour répartir les calculs équitablement au sein d'une grappe de serveurs.

Par la suite, nous allons nous concentrer en premier sur la version SQL principale des opérateurs, proposée dans la section (6.4), nous évaluerons ensuite les versions utilisant les astuces introduites dans la section (6.4.4), c.-à-d. le préfiltrage sur la signature des identifiants de classes pour l'opérateur $(-)$ et les fonctions de fenêtrage pour l'opérateur $(\wedge)$.

Les requêtes SQL correspondantes sont disponibles dans l'Annexe A. Tandis que des plans d'exécutions sont présentés pour chaque opérateur.

6.5.1 Configuration des expériences

L'ensemble de nos expériences est appliqué sur des partitions engendrées aléatoirement. Sachant deux partitions P et Q, nous allons évaluer les performances des opérateurs $P \wedge Q$, $P - Q$ et $P \vee Q$ uniquement. En effet, l'opérateur d'intersection $(\cap)$ dépend de l'opérateur de différence et est formulé comme le complément relatif à la différence. Seuls les opérateurs $\wedge$, $\vee$ et $-$ nous serviront de référence.

Concernant les expressions des requêtes SQL associées à chaque opérateur, des considérations empiriques ont été prises en compte. Suivant ce principe, nous avons entrepris certaines tentatives pour régler l'optimiseur de requêtes en tenant compte des différents algorithmes de jointures (Boucle imbriquée, hachage et tri) dans le SGBD que nous avons choisi.

Il s'avère que lorsqu'une amélioration significative peut être observée dans de nombreux tests, imposer une méthode de jointure peut mener à une dégradation brutale pour un même opérateur. Il semble donc que les plans d'exécution engendrés par l'optimiseur, sans imposer de contraintes, réalisent le meilleur compromis afin d'assurer des

performances équilibrées dans la plupart des cas de figure.

Les requêtes SQL ont été soigneusement conçues afin d'éviter les désagréments habituels (sous-requêtes inutiles, *etc*). Enfin, les plans d'exécution ont été examinés afin de repérer d'éventuelles stratégies sous-optimales.

Ayant auparavant exprimé le fait que le problème principal tient principalement au calcul de la fermeture transitive dans lequel les index ne seront d'aucun secours, nous avons néanmoins créé des index sur les identifiants des éléments et sur les ancres. D'autre part, nous avons évidemment prévu d'étendre les expressions atomiques à des requêtes plus complexes, faisant intervenir plus d'opérandes et mélangeant les opérateurs. Dans ce but, les algorithmes sont prévus pour fonctionner sans l'ajout d'informations auxiliaires.

Nous avons mené nos expérimentations sur un ordinateur doté de Windows 7 dont le processeur est un Intel Q6600 et cadencé à 2.4GHz. Les requêtes SQL ont été exécutées dans le SGBD PostgreSQL 9.1.

6.5.2 Générations des partitions

Dans ce qui suit, on définit type$(P) = \tau$, représentant la distribution des cardinalités des classes au sein d'une partition P et τ est une séquence décroissante où $\tau[i]$ donne la cardinalité de la classe a_i dans P.

Soit n objets, nous tirons aléatoirement, selon une loi de puissance, une partition P, puis nous engendrons une partition Q en appliquant des permutations sur P. En effet, une propriété remarquable de nombreux phénomènes naturels ou artificiels est que, étant donnée une population, la fréquence de ses sous-populations peut très souvent être modelée par une *loi de puissance* [32].

Générer de telles partitions est facilement réalisé en appliquant le processus (stochastique) du restaurant chinois (*Chinese Restaurant Process*) [58]. L'algorithme construit une partition aléatoire sur un univers d'entiers $[1, n]$. Un point remarquable mérite d'être soulevé : la distribution des cardinalités des classes, plus connue sous le nom de distribution d'Ewens, est dite interchangeable [1][107], c.-à-d. que la probabilité d'une partition dépend uniquement de la cardinalité de ses classes. Le nombre attendu de classes k croît suivant $O(\alpha \log n)$, où α est un facteur d'échelle de l'algorithme.

Ensuite, nous engendrons un ensemble de partitions $(P^{(1)}, \ldots, P^{(\ell)})$ en appliquant des permutations aléatoires sur P telles que :

- $d(P, P^{(i)}) < d(P, P^{(i+1)})$, suivant une distance $d : \Pi_\Omega \to \mathbb{N}$ qui évalue le nombre nécessaire de permutations à appliquer aux objets dans les classes de P pour atteindre sa contrepartie « bruitée » $P^{(i)}$;

[1] traduit directement depuis le terme anglophone (exchangeable).

- chaque $P^{(i)}$ conserve le type de P : $\mathsf{type}(P^{(i)}) \equiv \mathsf{type}(P^{(0)})$ lorsqu'on permute les objets des classes a_j et a_k ;

Le procédé de génération peut alors être résumé par les étapes suivantes :

1. Initialisation : choix aléatoire d'un ordre linéaire sur un sous-ensemble de classes $\{a_0, \ldots, a_\ell\} \subseteq P$;
2. $\forall i,\ 0 \leq i \leq \ell - 1$: permutation de $\min(\tau[i], \tau[i+1])$ objets entre chaque classe figurant dans la paire (a_i, a_{i+1}) afin de construire $P^{(i+1)}$.

Soit, par exemple, la partition $P = 1234|567|89$. P_1 sera construite en permutant les éléments des deux premières classes pour obtenir $P^{(1)} = 1567|234|89$. En répétant le processus depuis $P^{(1)}$, on obtient la partition $P^{(2)} = 1567|289|34$.

Une telle modélisation permet de contrôler précisément les opérations sur les partitions par rapport aux changements effectuées sur les opérandes, puis, déduire certaines propriétés pouvant avoir un impact sur les performances. Nous prenons note que l'identité suivante est toujours vérifiée :

$$\mathrm{rk}(P^{(0)}) = \mathrm{rk}(P^{(1)}) = \ldots = \mathrm{rk}(P^{(\ell)})$$

6.5.3 Résultats et analyses

Les résultats sont présentés pour des partitions définies sur $n = 1\,000$, $2,000$ et $5\,000$ objets. Bien que ces nombres soient assez modestes, nous avons observé dans nos expérimentations que le temps de traitement des requêtes nous empêchait d'augmenter significativement n.

Le nombre de classes est établi pour varier entre 10 et 20. Globalement, les plus grandes classes engendrées par l'algorithme sont typiquement de l'ordre de la moitié de l'ensemble total des objets à partitionner, tandis que pour $n = 2\,000$ and $n = 5\,000$, quelques singletons apparaissent en tant que classes.

Les temps relatifs à l'exécution des requêtes SQL sont reportés sur les figures (6.8), (6.9) et (6.10).

Les boîtes à moustaches décrivent les fluctuations observées dans un ensemble d'expérimentations où :

- Le nombre de classe varie dans l'intervalle $[10, 20]$;
- Le nombre de permutations aléatoires appliquées sur P pour engendrer chaque $P^{(i)}$ varie de 140 à 466 lorsque $n = 1\,000$, de 184 à 532 lorsque $n = 2\,000$, et de 406 à $1\,728$ lorsque $n = 5\,000$.

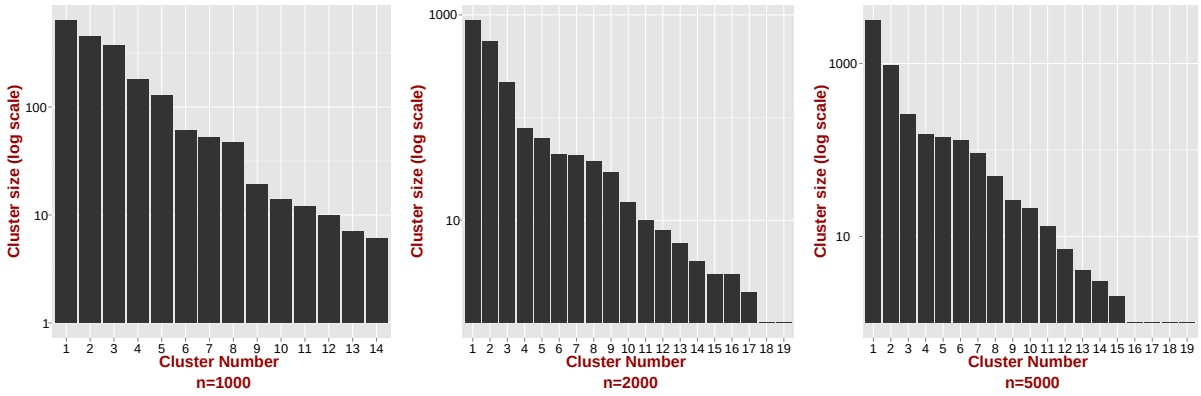


FIGURE 6.7 : Exemples représentatifs des distributions de cardinalités des classes avec $n = 1\,000, 2\,000, 5\,000$.

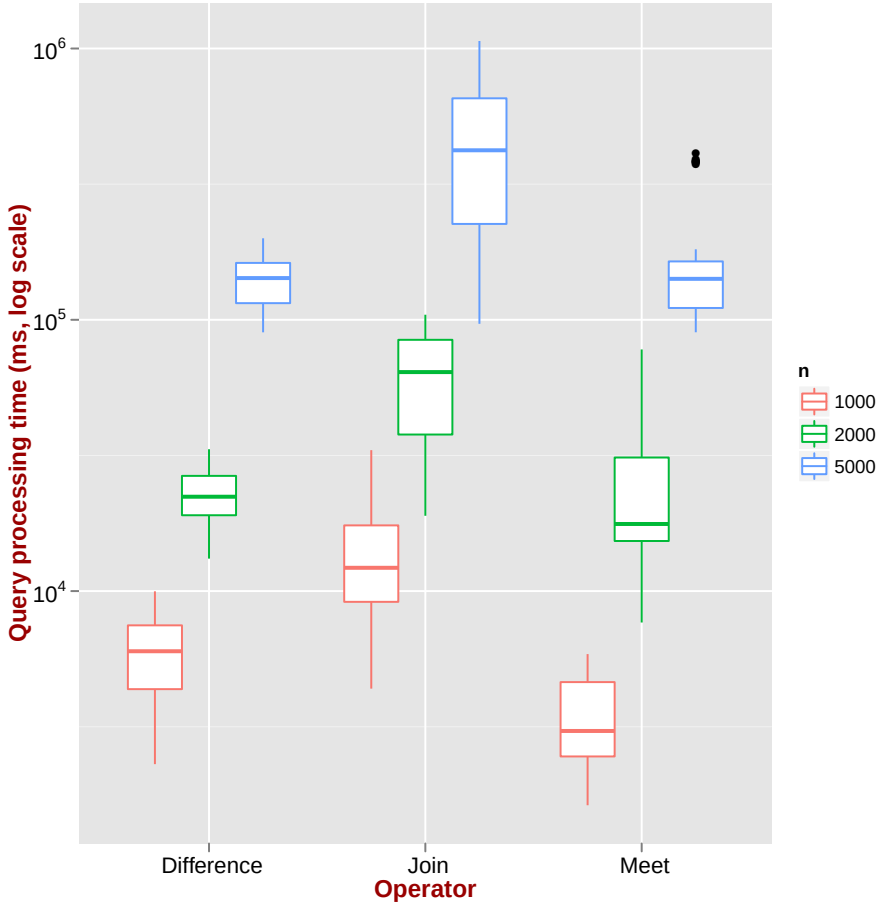


FIGURE 6.8 : Temps de traitement des requêtes SQL pour $n = 1\,000, 2\,000, 5\,000$ (pour les opérateurs principaux $-$,$\vee$ and $\wedge$, de gauche à droite).

Concernant les opérateurs principaux, l'opération $P \wedge Q$ est sensiblement plus coûteuse à réaliser que $P - Q$ tandis que les résultats du calcul $P \vee Q$ sont pires que tous les

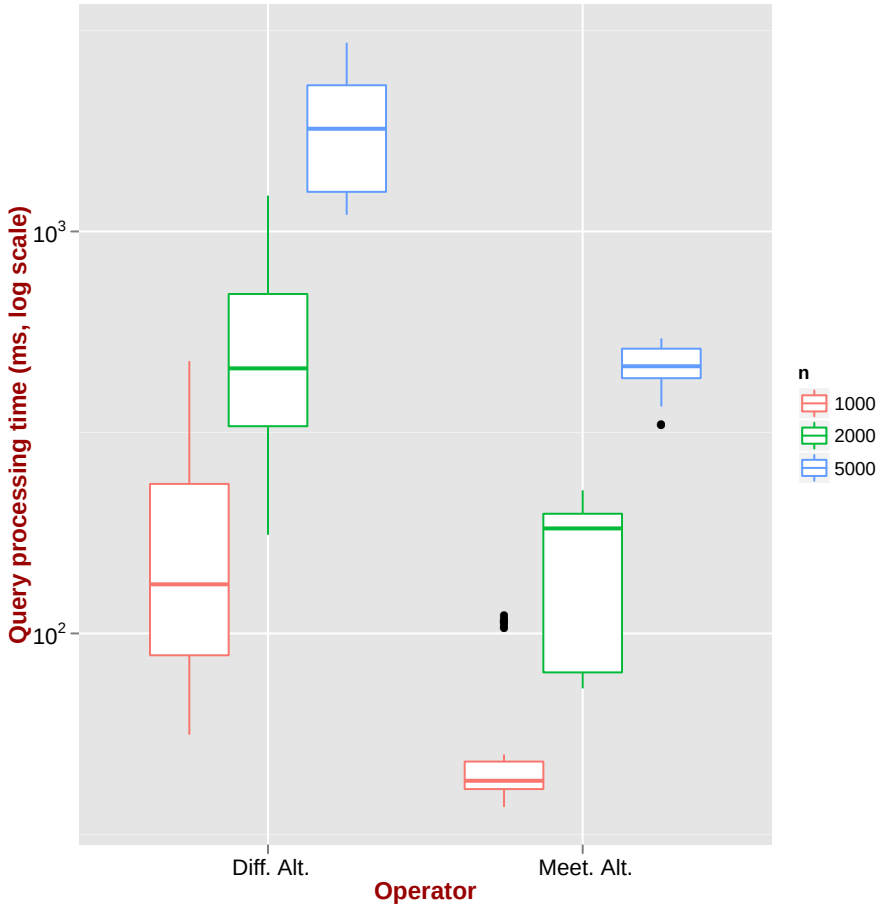


FIGURE 6.9 : Temps de traitement des requêtes SQL pour $n = 1\,000, 2\,000, 5\,000$ (pour les versions optimisées des opérateurs $-$ and $\wedge$).

autres. En outre, les opérations appliquées sur des partitions dont l'univers contient 5000 objets sont toutes plus coûteuses que celles avec 1000 ou 2000 objets ; le coût augmente d'un ordre de grandeur. De manière sommaire, ajouter du bruit dans les partitions accroît la moyenne quadratique du temps d'exécution.

À la fin, P op $P^{(\ell)}$, op $\in \{\wedge \vee -\}$ fait apparaître de nombreux cas aberrants à l'égard du temps d'exécution mesuré où $P^{(\ell)}$ est la plus lointaine partition de P. Cette observation est toutefois accentuée par l'augmentation du nombre d'objets n. La toute première conclusion pouvant être esquissée, est que le temps de traitement des requêtes devient rapidement rédhibitoire, même avec des jeux de données de taille modeste.

De plus, bien que les versions optimisées des opérateurs $(\wedge)$ et $(-)$, dont les temps de traitement sont reportés sur la figure (6.9) procurent des résultats meilleurs que leur principale version, on doit néanmoins constater que cela n'offre pas de contrepoids suffisant à la faible performance de l'opérateur $(\vee)$.

Nous avons évalué également dans quelle mesure, les versions optimisées pourraient être considérées comme un fondement solide et employées pour évaluer des requêtes

plus complexes. Suivant ce schéma, La figure (6.10) dépeint plusieurs tests indépendants où nous avons mesuré le temps d'exécution d'une séquence croissante de l'opérateur $(\wedge)$, dans sa version optimisée.

Initialement, le premier test ne contient que deux opérandes pour en atteindre dix, à la fin. Celles-ci sont aléatoirement choisies parmi l'ensemble $\{P^{(0)}, P^{(1)}, \ldots, P^{(\ell)}\}$. On peut observer un comportement strictement linéaire dans le temps de traitement. Il apparaît que l'optimiseur ne joue aucun rôle dans la manière dont sont gérées ces requêtes complexes, en dépit du fait que les partitions possèdent des propriétés algébriques propres à développer des stratégies d'optimisation.

En effet, puisque le rang des partitions reste toujours le même, cela implique que le résultat de $P \wedge Q$ induit toujours une partition dont le rang est toujours strictement inférieur aux opérandes impliquées. Puisque la distance entre les partitions de l'ensemble croît par rapport à la partition originale, $P^{(0)}$, il est attendu que le résultat espéré de n'importe quel test soit une unique opération $(\wedge)$ entre les deux partitions les plus éloignées parmi les opérandes, soit :

$$\bigwedge_{j=i}^{\ell} P^{(j)} = P^{(i)} \wedge P^{(\ell)}$$

Plus généralement, le calcul d'une séquence d'un même opérateur latticiel conduit à découvrir une formule équivalente qui combine le plus petit sous-ensemble de partitions afin d'atteindre la borne supérieure (respectivement inférieure) dans le treillis. Ce point souligne nettement la nécessité d'un plan d'exécution réfléchi et c'est là une question ouverte et une tâche ambitieuse.

6.6 Variantes pour le calcul de la borne supérieure

La motivation principale de cette section provient de l'incapacité à réaliser de manière optimale le calcul de la borne supérieure entre deux partitions. L'algorithme (6.4.1) calcule une relation d'équivalence sur les identifiants avant de réaliser une opération de réduction transitive afin de déterminer chaque nouvelle ancre de chaque nouvelle classe.

Cependant, puisqu'il existe un large choix de sous-graphes reliant les objets à unifier, dont le nouveau représentant, il est possible de réaliser la dernière étape sans qu'il soit nécessaire de construire la relation en entier. En particulier, nous cherchons à vérifier que le calcul du nouveau représentant est stable pour l'opération d'union entre ensembles. Par exemple, étant donnés deux ensembles X, Y telle que $Y = Y_1 \cup Y_2$, on vérifie l'identité suivante :

$$c(c(X \cup Y_1) \cup c(Y_2)) = c(c(X \cup Y_2) \cup c(Y_1))$$

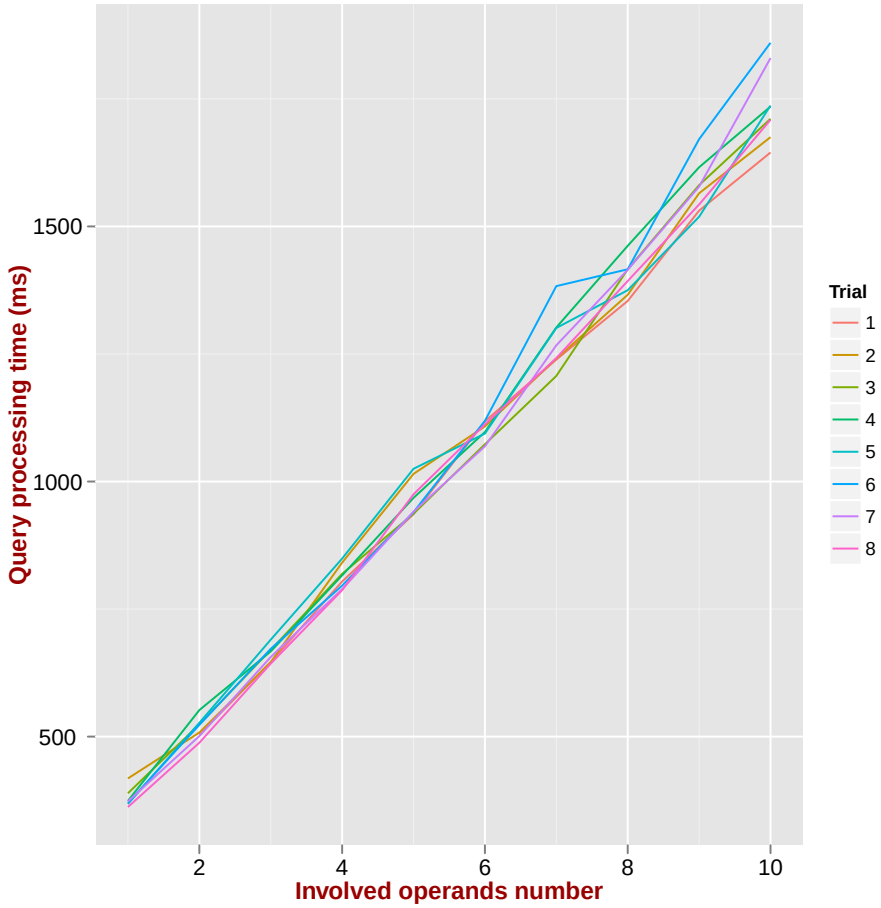


FIGURE 6.10 : Temps de traitement des requêtes SQL pour des séquences croissantes d'opérations de $(\wedge)$ utilisant la version fenêtrée avec $n = 5\,000$

qui se réécrit de manière fonctionnelle :

$$(c \circ \cup)(c(Y_2), (c \circ \cup)(X, Y_1)) \tag{6.11}$$
$$\iff (c \circ \cup)(c(Y_1), (c \circ \cup)(X, Y_2)) \tag{6.12}$$

Celle-ci souligne le fait que la fusion des classes X et Y est toujours indépendante de l'ordre choisi pour fusionner les sous-classes de Y. La décision finale sur le choix du représentant est donc indépendante de l'ordre de traitement des différentes parties et est donc compatible avec la commutativité et l'associativité de l'union des classes.

Il s'agit de démontrer dans un premier temps que le choix sur un ensemble est indépendant du choix sur ses parties.

Théorème 6.6.1. *Étant donné un ensemble $X \in \mathcal{P}(\Omega)$, il existe une décomposition $\mathcal{X} = \{X_1, X_2, \ldots\}$ telle que $X = \bigcup_{\mathcal{X}} X_i$ et vérifiant la propriété suivante :*

$$c(\bigcup_{\mathcal{X}} c(X_i)) = c(X) \tag{6.13}$$

Démonstration. Il est aisé de trouver une décomposition maximale $\mathcal{X}$ pour chaque ensemble en utilisant la carte $\phi(X) = \mathcal{J}(\mathcal{P}(\Omega)) \cap \downarrow X$, soit $\mathcal{X} \triangleq \phi(X)$. Puisque pour tout $\cup$-irréductible $X_j \in \mathcal{J}(\mathcal{P}(\Omega)) \implies c(X_j) = X_j$, on en déduit :

$$\bigcup_{\mathcal{X}} c(X_i) = \bigcup_{\mathcal{X}} X_i = X_i$$

ce qui complète la preuve. □

Ceci nous permet de définir une relation binaire définie sur $\mathcal{P}(\Omega) \times \mathcal{J}(\mathcal{P}(\Omega))$ dont chaque élément est un couple (X, x) qui s'interprète comme la classe X de représentant x où $x = c(X)$. Par exemple, étant donné $\Omega = \{1, 2, 3\}$, la figure (6.11) représente le demi-treillis $\mathcal{P}_{\min}(\Omega) \triangleq (\mathcal{P}(\Omega) \times \mathcal{J}(\mathcal{P}(\Omega)), \vee)$ associé.

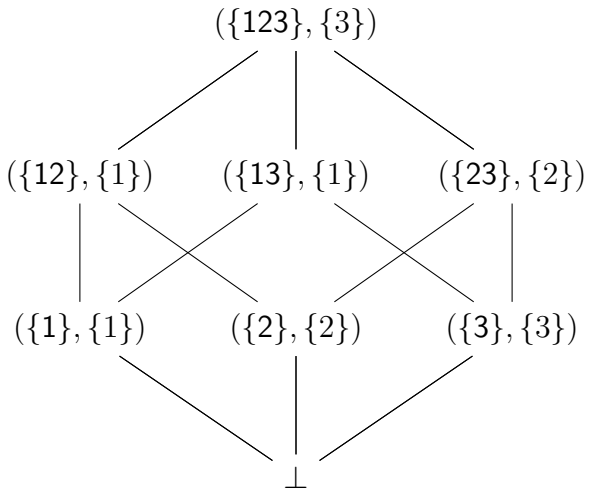


FIGURE 6.11 : Instances du treillis $\mathcal{P}_{\min}(\Omega)$

Étant donné $X, Y \in \mathcal{P}(\Omega)$, l'opération $(\vee)$ est définie de la manière suivante :

$$(X, \min(X)) \vee (Y, \min(Y)) = (X \cup Y, \min(\min(X) \cup \min(Y))$$

On va maintenant définir une relation d'équivalence sur les éléments de $\mathcal{P}(\Omega)$ qui partagent le même représentant. Cette relation est alors formée depuis les antécédents de la fonction $\min^{-1} : \mathcal{J}(\mathcal{P}(\Omega)) \to \mathcal{P}(\Omega)$.

Définition 6.6.1. *Soit $X, Y \in \mathcal{P}$, on définit la relation d'équivalence suivante :*

$$X \in [Y] \iff c(X) = c(Y)$$

Un corollaire évident de cette définition est que deux couples (X, x) et (Y, y) sont telles que $x = y$ si et seulement si $X \cap Y \neq \varnothing$. On en déduit le fait suivant.

Lemme 6.6.1. *Étant donné un ensemble $X \in \mathcal{P}(\Omega)$, il existe un ensemble $I \in [X]$ tel que pour tout $Y \in [X]$:*

$$Y \subset I \implies Y = I$$

I est alors un infinum de $[X]$.

Démonstration. Prouver cette propriété revient à vérifier que chaque Y dans la relation forme une famille $\mathcal{Y}$ telle que $\bigcap \mathcal{Y}$ soit dans l'ensemble. Or, l'intersection de $\mathcal{Y}$ est toujours définie et coïncide en $\bigcap_{Y\in\mathcal{Y}} \phi(Y) = \min(X)$ qui est l'unique point commun à tous les ensembles Y. □

Par la dualité de l'ordre, on obtient la propriété suivante :

Lemme 6.6.2. *Étant donné un ensemble $X \in \mathcal{P}(\Omega)$, il existe un ensemble $X \in [X]$ tel que pour tout $Y \in [X]$:*

$$Y \supset S \implies S = Y$$

S est alors un supremum de $[X]$.

Démonstration. Par la finitude de Ω, S est aisément constructible par la formule suivante :

$$S \triangleq \{y \in \Omega \mid \min(X) = \min(X \cup \{y\})\}$$

qui est donc l'ensemble maximal inclus dans $[X]$. □

Théorème 6.6.2. *Pour tout ensemble $X \in \mathcal{P}(\Omega)$, $([X], \subseteq, \cup, \cap)$ est un treillis complet où $\bot = \bigcap_{Y\in[X]}$ et $\top = \bigcup_{Y\in[X]}$.*

La preuve s'obtient aisément en démontrant que les opérations dans l'ensemble sont bien définies par l'existence de l'infinum et du supremum (*c.f.* également le théorème 3.2.2).

Nous allons maintenant démontrer l'assertion initiale.

Théorème 6.6.3. *Étant donnés les ensembles X et Y, telle que $Y = Y_1 \uplus Y2$, on vérifie alors la propriété suivante :*

$$\min(\min(X \cup Y_1) \cup \min(Y_2)) = \min(\min(X \cup Y_2) \cup \min(Y_1)) \tag{6.14}$$

Démonstration. Par le théorème (6.6.1), on sait que $\min(Y) = \bigcup_{\{y\}\in\phi(Y)}\{y\}$. Ceci implique que $\phi(Y_1) \cap \phi(Y_2) = \varnothing$ et seulement l'un des deux ensembles contient $\min(Y)$. On part du principe que $\min(Y) = \min(Y_1)$. On va maintenant montrer que chaque membre de l'égalité évalue nécessairement à $\min(X \cup Y_1)$; le résultat dépendant soit de X, soit de Y_1. Dans un premier temps, on constate aisément que :

$$\min(Y_1) \leq \min(Y_2) \implies \min(Y_1 \cup X) \leq \min(Y_2 \cup X) \tag{6.15}$$

En particulier, $\min(Y_1 \cup X) = \min(Y_2 \cup X)$ si et seulement si $\min(X) < \min(Y_1)$, mais dans ce cas, $\min(X \cup Y_1) < \min(Y_2)$. Du membre de gauche, on obtient facilement que :

$$\begin{aligned} & \min(\min(X \cup Y_1) \cup \min(Y_2)) && (6.16)\\ \iff & \min\min(X \cup Y_1) && (6.17)\\ \iff & \min(X \cup Y_1) && (6.18) \end{aligned}$$

Depuis le membre de droite, en utilisant la même relation, on obtient :

$$\begin{aligned} & \min(\min(X \cup Y_2) \cup \min(X \cup Y_1)) && (6.19)\\ \iff & \min(X \cup \min(Y_1 \cup Y_2)) && (6.20)\\ \iff & \min(X \cup Y_1) && (6.21) \end{aligned}$$

ce qui complète la preuve. □

Théorème 6.6.4. *Soit une famille d'ensembles $\mathcal{X} = \{X_1, X_2, \ldots\}$, la propriété suivante est vérifiée :*

$$\min(\bigcup_{\mathcal{X}} \min(X_i)) = \min(\bigcup_{\mathcal{X}} X_i) \tag{6.22}$$

Démonstration. On remarque facilement qu'au moins un $X \in \{X_1, X_2, \ldots\}$ est l'infinum de $[\bigcup X_i]$. En effet, on a :

$$\begin{aligned} & \min(X \cup \bigcup_{\mathcal{X}\setminus X} \min X_i) && (6.23)\\ \iff & \min(X \cup \bigcup_{\mathcal{X}\setminus X} X_i) && (6.24)\\ \iff & \min(\bigcup_{\mathcal{X}} X_i) && (6.25) \end{aligned}$$

□

Grâce à cette propriété, on voit que le choix du représentant peut se faire à n'importe quel moment de l'algorithme. L'application de l'opération $\min$ devrait alors être réalisée le plus tôt possible dans l'algorithme calculant $\varepsilon(P \vee Q)$.

Cela soulève une question : sous quelles conditions peut-on appliquer une telle substitution dans le but de réduire le coût du traitement et donc, le nombre de tuples à ajouter dans la relation θ ? Comme nous venons de le rappeler, notre approche en deux étapes est totalement formulable à l'aide d'une requête récursive en Datalog. Les tuples collectés dans θ sont pour la plupart redondants et ceux partageant le nouvel identifiant de la classe nécessitent de traiter chaque réaffectation, l'une après l'autre.

Parcourir θ ne peut être évité pour réaliser la mise à jour, en particulier lors de la dernière étape où $|\theta| = O(|\Omega|^2)$. Puisque cette relation est facilement manipulable par un graphe orienté binaire, c'est la direction que nous emprunterons par la suite.

Enfin, la combinaison des couples dans $\mathcal{P}_{\min}(\Omega)$ n'est pas gratuite : celle-ci est subordonnée à l'existence d'une classe intermédiaire dans la relation encodant la seconde partition qui peut être visitée plusieurs fois ; représentant une augmentation non-négligeable du coût de traitement. Nous verrons en particulier quelles stratégies peuvent être adoptées pour minimiser ce coût.

6.6.1 Structure de données et problématique associée

Nous allons décrire dans cette partie la structure de données que nous allons utilisée pour s'affranchir des limitations constatées dans notre précédente implémentation. En outre, celle-ci suggère la représentation à adopter pour le stockage des relations associées aux partitions.

Le principe selon lequel nous fondons notre solution, consiste en la mise en œuvre de l'opération d'unification des classes par l'instanciation d'une structure de données *Union-Find*[2] [69, 129]. Suivant ce cadre, on a à notre disposition les trois fonctionnalités suivantes :

- ens(x), engendre un singleton contenant x ;
- classe(x), retourne le représentant de la classe dont fait partie x. Après ens(x), on aura alors classe$(x) = x$. Dans le cas contraire, la recherche du représentant passe par l'entremise d'un ou plusieurs objets, reposant entre x et le représentant ;
- union(x, y), comme le nom le suggère, unifie les classes associées x et à y. Le choix du représentant relève habituellement d'un critère statistique, indépendant de Ω et sur lequel nous reviendrons.

Un tel cadre algorithmique est le plus souvent employé dans la *construction* et la *maintenance* de composantes connexes dans un graphe, c.-à-d. en utilisant des structures de nature arborescente afin d'épargner de la ressource mémoire et de diminuer le temps de traitement. D'autres exemples d'applications sont disponibles dans l'étude de Galil et Italiano [56].

Tarjan [129] donna une première borne sur la complexité en temps nécessaire pour réaliser un certains d'opérations d'union et de classe. Tandis que dans Fredman [55], il est montré qu'un tel algorithme est asymptotiquement linéaire sur le nombre d'opérations totales à réaliser. Partons du principe que nous avons n union à réaliser, qui nécessitent m classe, alors on a le résultat suivant :

Théorème 6.6.5. *La complexité de l'algorithme utilisant les heuristiques d'*union par rang *et de* compressions de chemin *se réalisent en* $O(n + m \cdot \alpha(m, n))$.

La fonction $\alpha(.,.)$ désigne l'inverse de la *fonction d'Ackermann* qui est largement connue pour sa croissance fulgurante :

$$\begin{cases} A_0(x) = x + 1 \\ A_{k+1}(x) = A_k(x) \end{cases}$$

Tandis que $\alpha(n)$ se définit alors comme la fonction minimisant k sous la contrainte

[2] Nous préférons l'appellation anglophone plutôt que « classe-union », moins usitée.

$n \leq A_k(2)$. Les premières valeurs de la fonction sont alors :

$$
\begin{aligned}
A_0(2) &= 3 \\
A_1(2) &= 4 \\
A_2(2) &= 8 \\
A_3(2) &= 2048 \\
A_4(2) &= \underbrace{2^{2^{2^{\cdot^{\cdot^{\cdot}}}}}}_{2048}
\end{aligned}
$$

où le dernier terme est supérieur au nombre d'atomes estimé dans l'univers (10^{80}). Ceci permet de supposer qu'aucune exécution de l'algorithme ne sera susceptible de dépasser $\alpha = 4$, et pour n suffisamment grand, l'algorithme sera *presque* linéaire.

Des différences notables apparaissent entre les contraintes imposées par notre problème et la solution proposée dans ce cadre.

Le premier point est que nous ne pouvons pas formuler d'hypothèse sur la structure des classes que nous souhaitons fusionner (densité, voisinage, *etc*.) et donc des *dépendances* entre les objets classés différemment dans les deux partitions. En effet, en appliquant l'algorithme classique, nous obtiennons alors un nombre maximal d'union $n = |\Omega^2|$ et donc une complexité en $o(n^2)$, ce que nous cherchons précisément à éviter.

Puisque nous ne pouvons pas connaître en avance le nombre minimum de liens à maintenir entre les objets, nous devons restreindre ce nombre de sorte que l'étape de réduction reste applicable et efficace lors de chaque phase du traitement.

Techniques de fusion des classes

Le second point est que l'application d'une union matérialise l'instance résultante de la fusion des classes associée aux paramètres. Il est généralement entendu que cette opération doit être réalisée en temps constant $O(1)$. Un critère statistique simple est généralement utilisé afin de ne pas déséquilibrer la nouvelle structure : on détermine le nouveau représentant tel que la hauteur maximale de l'arbre reste inchangée (union par rang).

Dans notre cas, la décision dépend de l'identifiant attaché à chacun des représentants des deux classes et ne permet pas *a priori* d'optimiser l'organisation des nœuds dans la structure. Suivant l'ordre de traitement des objets, c'est susceptible d'agraver les performances.

Pour illustrer ce problème, prenons par exemple les partitions $P = 1234|56$ et $Q = 12|3456$. Au départ du traitement, une représentation des classes associées au résultat est la suivante :

Après plusieurs étapes, une partie des objets est associée dans une classe, comme observé sur la figure (6.13) :

FIGURE 6.12 : Initialisation des classes de la partition $P \vee Q$

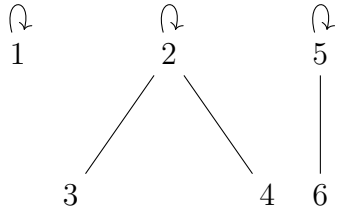


FIGURE 6.13 : Partition en cours de construction $1|234|56 \leq P \vee Q$

Sachant que les objets 1 et 4 devront être fusionnés l'un et l'autre, un appel à $\mathsf{union}(\mathsf{classe}(1), \mathsf{classe}(4))$ produirait dans le cas classique le résultat de la figure (6.14) :

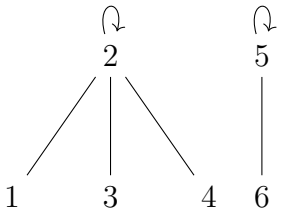


FIGURE 6.14 : Fusion des classes $\{1\}$ et $\{234\}$

En effet, puisque la hauteur de la structure associée à la classe de 2 est plus haute que celle de 1, l'ajouter en tant que nœud fils, n'agrandit pas la structure. Des appels subséquents à classe ne verront pas leur temps de traitement augmenter. Par opposition, nous sommes contraints par notre schéma d'encodage à placer 1 comme nouveau représentant de la classe ; comme indiqué sur la figure (6.15) :

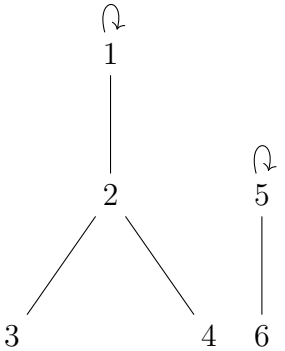
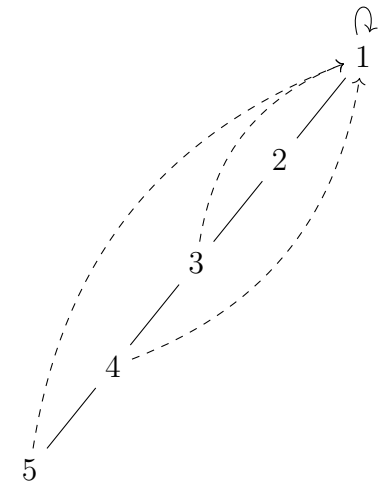


FIGURE 6.15 : Fusion des classes $\{1\}$ et $\{234\}$

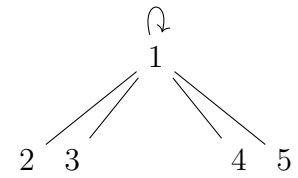
Donc le choix du représentant diminue le nombre d'alternatives permettant de maîtriser la profondeur des arbres, et dont dépend la complexité de la fonction classe. Elle

présente néanmoins l'avantage de permettre l'application de la réduction transitive le plus tôt possible.

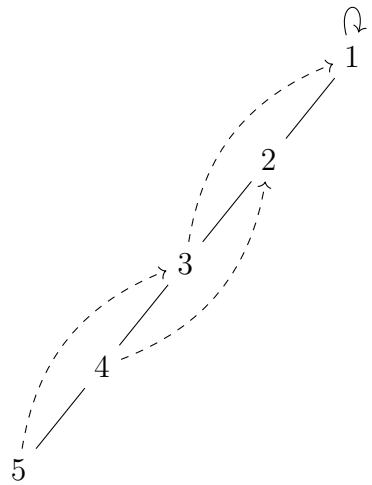
Techniques d'application de la réduction transitive



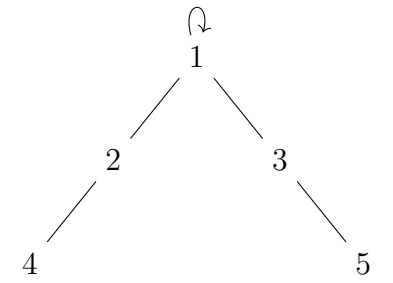
(a) Une classe avant la compression de chemin induite par classe(5)



(b) La même classe après la compression de chemin



(c) Une classe avant la compression de chemin par séparation induite par classe(5)



(d) La même classe après la compression de chemin

FIGURE 6.16 : Application des deux méthodes de compression de chemin

Généralement, la réduction transitive est appliquée lors de la recherche de la racine d'une classe en redirigeant tous les nœuds se trouvant le long du chemin vers la racine, et sans remettre en question la cohérence de la classe. La figure (6.16) présente les deux techniques les plus courantes pour rattacher les nœuds vers leur racine.

La première technique, figurée par (6.16(a)) nécessite deux passes pour réaliser l'opération : on remonte une première fois vers la racine, une fois celle-ci déterminée, on repart depuis l'élément passé en paramètre afin de rattacher tous les nœuds vers leur racine.

La second technique, figurée par (6.16(c)), est couramment désignée par *path splitting* car elle réalise une bipartition des nœuds le long du chemin. Elle applique ensuite sur chacune des parties, la redirection vers la racine.

Il est important de noter que ces deux techniques, lorsqu'elles sont combinées avec une technique d'union des classes, possèdent la même borne asymptotique énoncée dans le théorème (6.6.5).

Par ailleurs, d'autres combinaisons d'approches existent mais nous ne les énumérerons pas toutes dans cette partie, celles-ci ayant un intérêt limité dans notre cas.

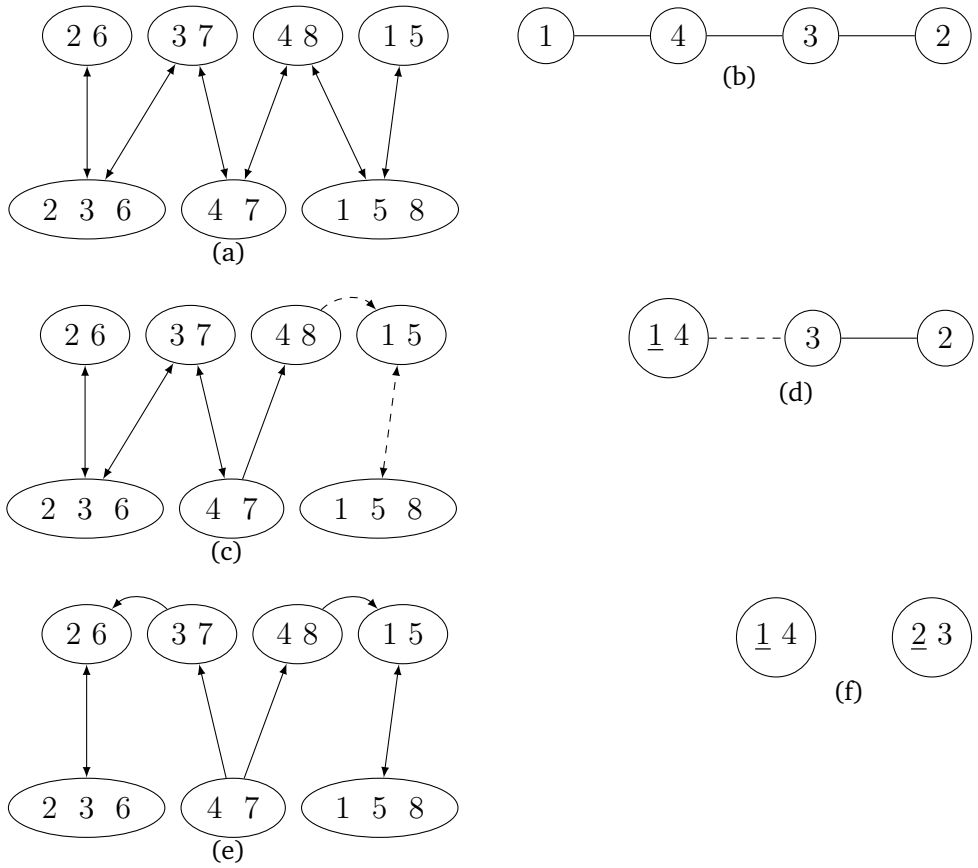


FIGURE 6.17 : Illustration des intersections entre les classes de P et de Q. À droite, chaque nœud du graphe est identifié avec le représentant de la classe de P.

Une manière de caractériser le point qui les unit est de considérer l'appel permettant de les réaliser : union(classe(x), classe(y)). Toutes les méthodes ont en commun qu'elles appliquent la réduction transitive depuis x et depuis y vers leur classe respective. Or, lors d'une opération d'union, l'ordre de traitement sur les identifiants de classe possède

un impact significatif sur l'obtention du résultat.

Posons les partitions $P = 15|26|37|48$ et $Q = 158|236|47$, la figure (6.17(a)) montre l'ensemble des chemins passant par les classes de Q et qui permettent de réaliser l'union des classes de P.

L'ensemble des unions à réaliser, telles qu'illustrées par la figure (6.17(b)), nécessite donc la complétion de l'ensemble[3] suivant :

$$\mathcal{P}_{\min}(\Omega) = \{(15, 1), (26, 2), (37, 3), (48, 4)\}$$

On présume qu'un algorithme parcourt les classes dans l'ordre suivant $(1, 2, 3, 4)$. Naturellement, puisque $15 \cap 158$ et $158 \cap 48$, une seule opération d'union doit être réalisée. Celle-ci renvoie alors le couple $(1548, 1)$.

Or, l'opération a un effet indésirable sur la suite du traitement : sur la figure (6.17(d)) représentant la nouvelle affectation de 4 à 1, la ligne pointillé illustre le fait que si la classe 3 est accessible depuis 4, elle ne l'est pas depuis $1 \equiv 4$. Bien que la classe 4 n'ait pas encore été traitée, $\text{classe}(4) = 1$ ne permettra pas de découvrir l'intersection avec la classe $\{4, 7\} \in Q$. Ce fait est illustré sur la figure (6.17(c)) où seul un arc entrant permet d'atteindre la classe 4 puis 1.

Si on continue le traitement sur les classes restantes $(2, 3, 4)$, le même problème survient de manière symétrique. Cette fois-ci, les classes 3 et 4 ne sont plus joignables en raison de leur rattachement aux classes 2 et 1, comme le montre la figure (6.17(e)).

Plusieurs solutions existent pour résoudre le problème : il s'agit en premier lieu de garantir une couverture des classes de P par l'entremise des classes de Q avant l'application de la fermeture transitive. La première solution est de trier suivant un ordre quelconque $(1, 2, 3, 4, \ldots)$ les représentants de classes de P, puis n'admettre l'union qu'avec des classes de P, déjà traitées.

Si on repart de l'exemple précédent, le traitement des classes 1 et 2 n'aura pas d'effet sur la structure, étant donné que 1 est le plus petit entier identifiant une classe. Inversement, la classe 2 peut être liée uniquement qu'avec des classes dont l'identifiant est plus grand.

La figure (6.18) montre l'élaboration de l'union des classes par le traitement des classes 3 et 4. Les deux premières étapes consistent en l'union des couples $(37, 3) \vee (26, 2) = (2367, 2)$ et $(48, 4) \vee (15, 1) = (1458, 1)$. Bien que le résultat soit similaire d'un point de vue des classes, les classes 4 et 3 restent mutuellement accessibles afin de finir l'algorithme par $(1458, 1) \vee (2367, 2) = (12345678, 1)$.

Une autre possibilité est d'*entrelacer* l'opération de classe avec l'opération d'union. Celle-ci a pour particularité de ne pas relier les classes par leur représentant mais de mélanger les ordres. L'avantage de cette approche est de profiter de l'ordre induit par chaque classe. C'est le choix que nous avons retenu pour mettre en œuvre nos algorithmes.

[3] afin de simplifier la lecture, les ensembles sont mis à plat tels que, par exemple, $(26, 2)$ est équivalent à $(\{2, 6\}, \{2\})$

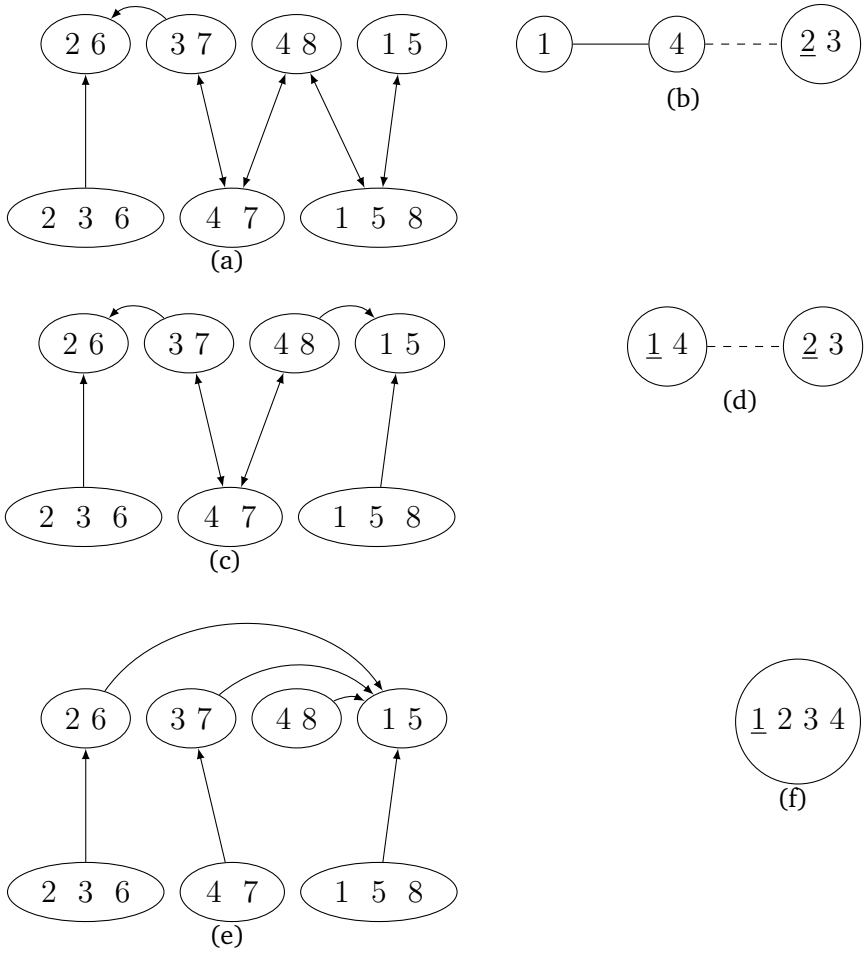


FIGURE 6.18 : Illustration des intersections entre les classes de P et de Q. À droite, chaque nœud du graphe est identifié avec le représentant de la classe de P.

Nous proposerons également une variante ne faisant pas l'usage d'un ordre sur le traitement des classes mais réalisant l'opération en deux passes. La première passe est dévolue à la réalisation d'union fiable entre classes – interdisant la fermeture transitive. Dans le cas contraire, toute classe dont l'union avec une autre est marquée afin d'être traiter dans la seconde.

6.6.2 Tri préalable des classes

La première solution que nous souhaitons présenter est également la plus simple : l'ensemble des classes est trié dans l'ordre *croissant* afin de maîtriser la propagation des unions de classe.

Ceci évitera que θ contienne la relation d'équivalence intégrale et permet au contraite de matérialiser uniquement un sous-graphe couvrant, offrant l'accessibilité depuis n'importe quel nœud vers le représentant.

Nous allons dans un premier temps réutiliser l'approche employée pour parcourir les relations associées à chaque partition de la section (6.4.3). La formule r_2 crée des dépendances entre les classes de M dès lors qu'il existe un chemin traversant une classe de N, et permet de réétiqueter toutes les classes associées dans le résultat de $\varepsilon(P \vee Q)$.

De plus, les représentants associés aux objets du domaine actif sont tous modélisés à l'aide de pointeurs. Cela nous permet de changer le représentant d'une classe par effet de bord, et ce en temps constant. En particulier, grâce à cette représentation, l'initialisation suivante :

$$\theta \leftarrow \{(x, x) : \exists y.M(x, y)\}$$

permet de nous préoccuper de la fusion des classes. Elle est équivalente à la création d'une partition $\perp$ où chaque objet est un identifiant de classe.

Algorithme 6.6.2 Version améliorée de $\varepsilon(P \vee Q)$

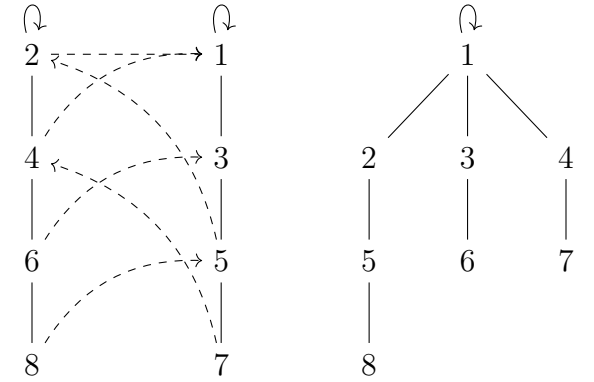
```
1: θ ← {(x,x) | x ∈ π2(M)}
2: tri croissant θ                                   ▷ Étape d'initialisation
3: pour tout (x,x) ∈ θ faire
4:     pour tout u | ∃t (M(t,x), N(t,u)) faire
5:         si b(u) existe alors
6:             y ← b(u)
7:             interrompre
8:         sinon
9:             pour tout y | ∃v (N(v,u) ∧ M(v,y)) faire
10:                si x > y alors
11:                    ENTRELACEMENT(x,y)
12:                    CREATION_PONT(p(x),p(y),u)
13:                    ENTRELACEMENT(b(x),u)
14:                    b(u) ← x
15:                fin si
16:            fin pour
17:            y ← ∃v (N(v,u) ∧ M(v,y))
18:        fin si
19:    fin pour
20: fin pour
```

L'algorithme (6.6.2) suit ces principes. Les appels à la procédure récursive ENTRELACEMENT réalise l'union des classes associées à chaque paire d'identifiant de classes (x, y). Pour réaliser cela, on dispose de la fonction $p(.)$ qui associe à chaque identifiant x, son parent dans la structure associée à la classe. En effet, contrairement à la procédure d'union classique, il n'est pas nécessaire de disposer du représentant renvoyé par classe(x) pour relier les classes entre elles.

Par ailleurs, chaque classe de la relation N étant susceptible d'être visitée plusieurs fois, l'algorithme maintient une structure dynamique entre les classes de chaque relation afin de ne pas balayer les objets des classes u plus d'une fois. En particulier, cette structure nous permet également de réaliser l'union des classes de la relation N.

Union par entrelacement

Pendant le déroulement de l'algorithme, lorsqu'une paire d'identifiants (x, y) de la relation M est telle que leurs classes respectives doivent être fusionnées, l'appel à la procédure ENTRELACEMENT réalisent la fusion des deux classes en mélangeant les ordres propres à chacune des classes (*c.f.* [39, 128]). Une illustration de cette union est exposée sur la figure (6.19).



(a) Deux classes avant entrelacement induit par union$(8, 7)$

(b) La classe résultante

FIGURE 6.19 : Fusion des classes par union des ordres $(7, 5, 3, 1)$ et $(8, 6, 4, 2)$

Ayant l'avantage de disposer d'un ordre sur les identifiants de classes, l'union des ensembles doit donc préserver cet ordre. Pour chaque nœud, pris à tour de rôle dans chaque classe, on cherche le *premier élément* dans la seconde classe qui soit immédiatement inférieur à lui et on le relie. La construction de la structure est détaillée par la figure (6.20) et est résumée dans l'algorithme (6.6.3).

Algorithme 6.6.3 Entrelacement des classes x et y

```
1:  procedure ENTRELACEMENT(x, y)
2:      si p(x) ≠ p(y) alors
3:          si p(x) < p(y) alors
4:              tmp ← p(y)
5:              y ← tmp
6:              p(y) ← p(x)
7:          sinon
8:              tmp ← p(x)
9:              x ← tmp
10:             p(x) ← p(y)
11:         fin si
12:         ENTRELACEMENT(x, y)
13:     fin si
14: fin procedure
```

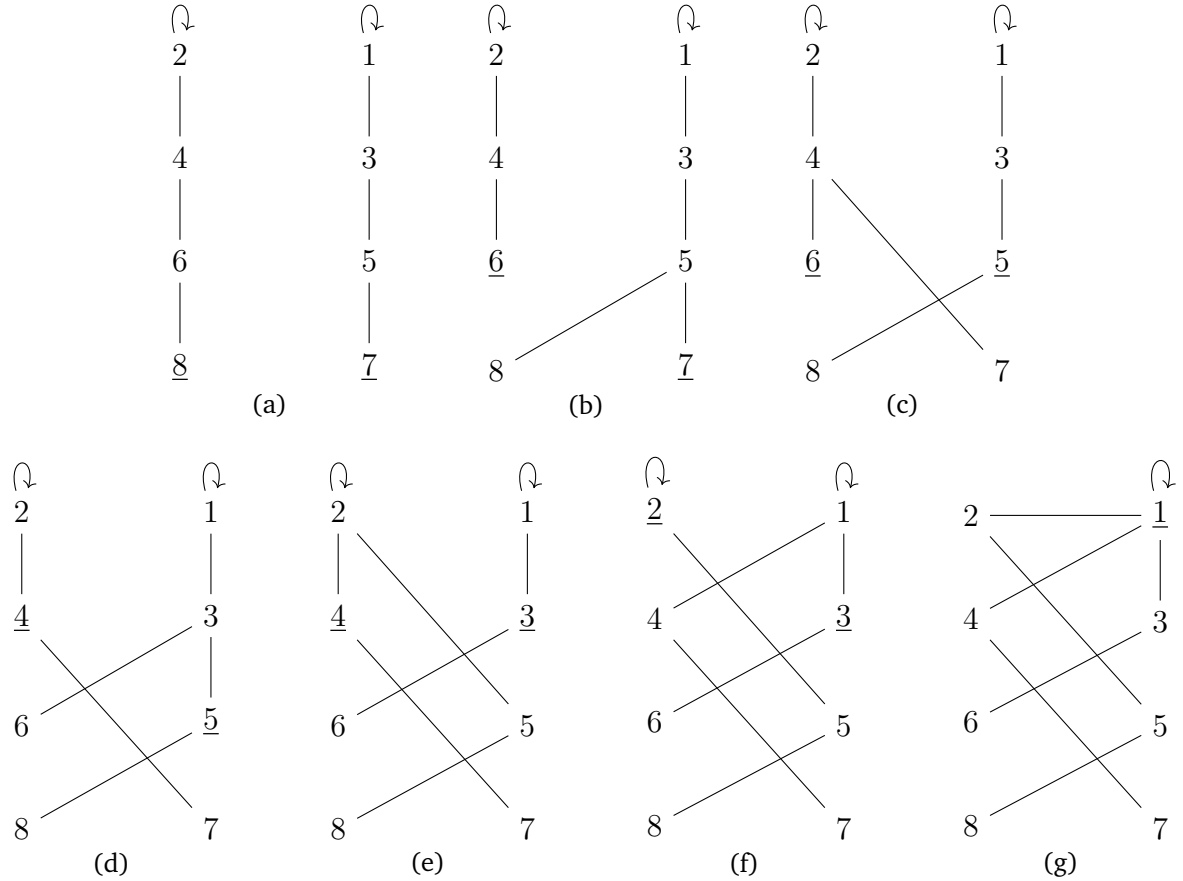


FIGURE 6.20 : Ensemble des opérations de rattachement réalisées lors de l'appel à union$(8, 7)$. Les identifiants soulignés forment le couple (x, y) de chaque appel récursif de la procédure.

C'est particulièrement intéressant car lorsque le nombre d'éléments dans une classe est restreint et que son représentant est très grand, unifier ne nécessite pas de remonter jusqu'en haut de l'autre structure et réduit donc le temps de calcul nécessaire. En pratique, nous n'avons effectivement jamais observé de tels cas extrêmes : quelle que soit le jeu de données considéré, la profondeur des arbres n'excède jamais 2 ou 3. De plus, l'utilisation de cette heuristique a permis de réduire de plusieurs ordres de grandeur (passant de 10^6 à 10^2 opérations sur des partitions de $100\,000$ objets, en moyenne), le nombre total de remontées dans les structures lors des différents essais, prouvant ainsi son utilité.

Passerelle entre les relations M et N

Une particularité de l'algorithme exploite le fait que le calcul $\varepsilon(P \vee Q)$ est effectué en complétant P – dans sa version relationnelle M –, par la recherche du moindre point-fixe commun aux deux partitions. Or, rien n'empêche de réaliser ceci à la fois sur M, mais aussi sur N. En effet, construire les couples $(x, y) \in \pi_2(M)$ nécessite de parcourir

entièrement les objets de la même classe que celle de x et rechercher une intersection avec une classe de N, dont l'identifiant est stockée dans la variable u de l'algorithme (6.6.2).

Algorithme 6.6.4 Initialisation de la relation entre les classes de M et de N

```
1:  procedure CREATION_PONT(x, y, u)
2:      si b(x) et b(y) n'existe pas alors
3:          b(y) ← u
4:          b(x) ← u
5:      sinon si b(x) n'existe pas alors
6:          b(x) ← b(y)
7:      sinon si b(y) n'existe pas alors
8:          b(y) ← b(x)
9:      fin si
10: fin procedure
```

Nous maintenons donc un graphe biparti, relation binaire construite pendant le déroulement de l'algorithme, entre les classes de M et celles de N. Lorsqu'une intersection (x', u) est découverte de part et d'autre des deux relations, si un couple (x, u) a amené précédemment à la fusion d'un couple (x, y) alors on doit également fusionner, soit les classes de (x', x) ou bien de (x', y). Posons par exemple, les partitions $P = 123|456|789$ et $Q = 147|258|369$. On vérifie aisément qu'elles sont complémentaires :

$$P \wedge Q = \bot$$
$$P \vee Q = \top$$

La première équation indique que chaque cluster de P intersecte chaque cluster de Q et permet de construire un graphe biparti complet. À l'opposé, la seconde indique que toutes les classes de chaque partition doivent être fusionnées pour construire le résultat. La figure (6.21) présente la construction pas-à-pas de l'union des classes des relations associées à P et à Q.

L'algorithme va chercher à fusionner les classes de P dont les points de départ sont les représentants $\theta = \{(7, 7), (4, 4), (1, 1)\}$. La première étape, représentée par la figure (6.21(a)), réalise la construction de la structure en partant depuis le représentant de la première classe à traiter. La figure (6.21(b)) présente la seconde étape où un second pont est réalisé avec les deux premières classes de P. Incidemment, cela permet l'union entre les deux classes de Q, et seul le premier pont est alors conservé. La visite depuis le dernier élément permet l'union de la dernière classe de Q (Fig. 6.21(c)) tandis que le domaine actif de chaque partition a été parcouru une seule fois. Aucun nouveau pont n'est donc retenu à la fin de cette étape.

Le traitement des deux autres classes de P est court-circuité, étant donné le pont entre elles et les autres classes de P, et permettant de terminer l'algorithme. À la fin, la réaffectation des identifiants de classe retourne $P \vee Q = \top$.

Cette heuristique de choix dépend des lignes 5 à 8 dans l'algorithme. Dans ce cas,

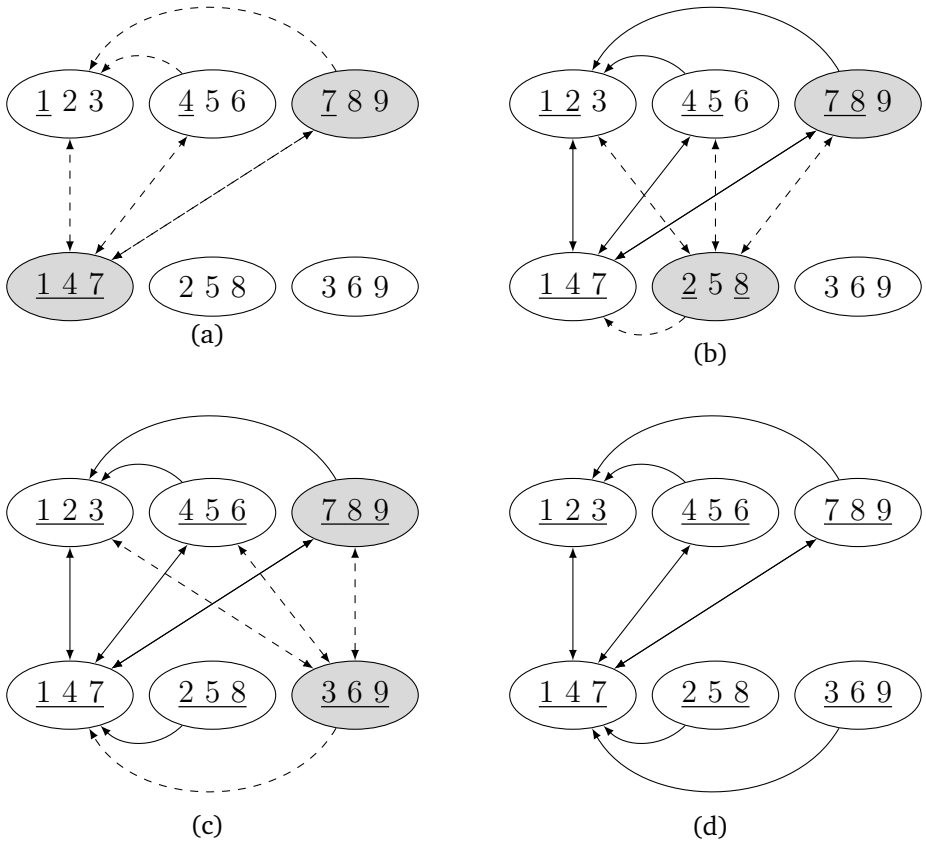


FIGURE 6.21 : Illustration de la construction et du fonctionnement du graphe entre les relations M et N pour le calcul de $\varepsilon(P \vee Q)$. Les membres soulignés sont ceux déjà visités. Les liens tracés en pointillés sont ceux construits au cours de l'itération. Les classes grisées sont celles visitées durant la même itération.

l'appel de la fonction $b(u)$, ligne 6, retourne directement l'identifiant de la classe y avec laquelle la classe x doit fusionner.

Une fois réalisées les optimisations sur M, le pont réalisé entre les deux relations permet d'appliquer le même algorithme en fusionnant cette fois-ci les classes de N. Cette dernière opération a pour avantage de réduire le nombre d'intersections à découvrir sur les classes de N et donc de diminuer le nombre d'éléments à parcourir dans les classes de N.

À la fin de l'algorithme, θ ne contient que les réaffectations à réaliser pour les seules classes de M. La même opération peut être également réalisée concomitamment sur N bien que cela ne présente pas d'intérêt particulier.

6.6.3 Version stochastique

Lorsque le tri des identifiants de classes de M n'est pas réalisable soit pour des contraintes de ressources mémoire ou de coût de traitement, deux options peuvent être réalisées pour contourner le problème :

- Adapter la structure de données maintenant les arbres de sorte à obtenir une structure de données persistante (c.-à-d. munie de la fonction « défaire ») telle que les unions de classes puissent être annulées avec un coût minimum [36] ;
- Retarder les décisions relatives, soit à l'union des classes, soit à la réduction transitive

La seconde possibilité semble plus aisée à mettre en œuvre car elle concerne uniquement la modification des règles d'unions des classes. Dans les deux cas, la passerelle maintenue entre les relations dans la version déterministe de l'algorithme que nous proposons n'est plus pertinente car elle dépend de la capacité à appliquer un changement qui ne puisse être repudié après coup.

Comme mentionné plus haut, l'infaillibité d'une décision dépend de l'ordre de traitement des identifiants de classes de M. Chaque identifiant étant traité *une seule* fois, nous devons garantir que chaque décision liée à un autre identifiant – qui n'a pas encore été traité – sera repoussée si jamais une opération de réduction transitive doit être appliquée. Ceci est le point saillant de l'algorithme (6.6.5) et est figurée par les lignes $8, 13$ et 20.

Lors de la première passe, chaque classe dans θ ne peut fusionnée qu'avec une autre classe si cela n'implique pas de fermeture transitive sur une classe qui n'a pas encore été traitée. Dans le cas contraire, si aucun traitement n'est permis lors de la première passe, son application est différée lors de la seconde passe. Cette sécurité prévient que chaque identifiant nouvellement traité ne sera redirigé vers son représentant, uniquement dans le cas où un chemin alternatif dans le graphe existe par l'entremise d'un autre identifiant. La réduction transitive est ensuite pleinement réalisée lors de la seconde passe de l'algorithme. Ainsi, les arcs qui sont les seuls chemins disponibles ne seront pas supprimés par inadvertance par l'algorithme.

Reprenons les partitions $P = 15|26|37|48$ et $Q = 158|236|47$; on suppose l'ordre suivant sur les identifiants $(4, 3, 1, 2)$. On tire donc l'identifiant 4, et on constate que sa classe peut fusionner avec les classes 1 et 3, sans restriction sur l'ordre de traitement. On présume que l'on doive réaliser $(48, 4) \vee (36, 3) = (3468, 3)$. Dans ce cas, étant donné que $\mathsf{classe}(4) = 3$, on rejette l'opération d'union avec 1 car celle-ci implique le traitement de 3 *via* la fermeture transitive ; la classe 4 est alors marquée et sera de nouveau sollicitée dans la seconde passe. À la fin du traitement de 4, seules les classes 3 et 4 ont fusionné ensemble, comme le montre les figures (6.22(a)-6.22(b)).

Depuis la classe 3, on ne peut donc accéder qu'à la classe 2, dont on réalise la fusion, ainsi que le montre les figures (6.22(c)-6.22(d)). Le traitement des classes 2 et 1 ne

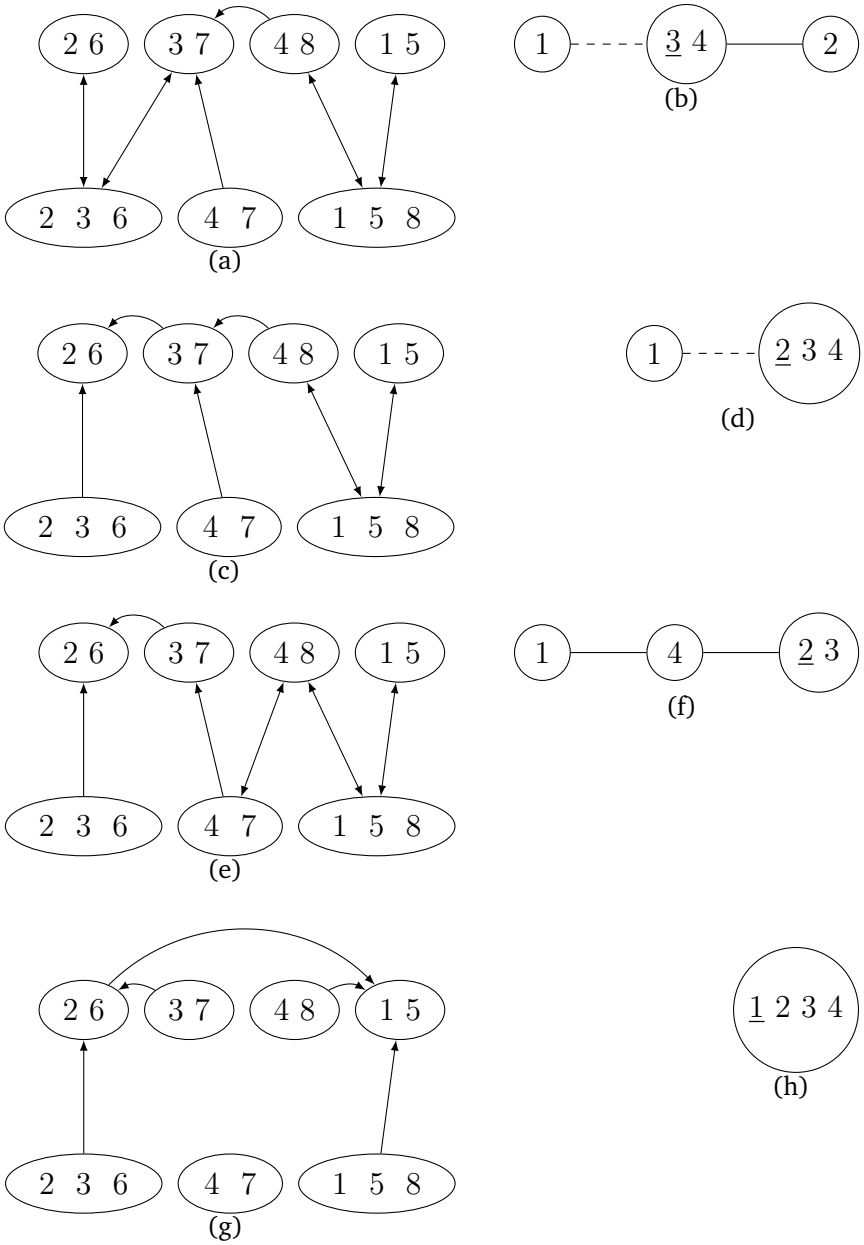


FIGURE 6.22 : Illustration des intersections entre les classes de P et de Q. À droite, chaque nœud du graphe est identifié avec le représentant de la classe de P.

provoque pas de changement puisque ces classes ne peuvent être fusionnées avec une autre classe. À la fin de la première passe, on a donc $\theta = \{(4, 3)\}$. Afin de traiter de nouveau la classe 4, celle-ci est réinitialisée : on peut dès lors accéder à l'ensemble des classes devant être fusionnées comme l'indique les figures (6.22(e)-6.22(f)). Agissant sans restriction, on peut fusionner le reste des classes ensemble.

Il s'ensuit que si le graphe est au moins régulier de degré 2, alors il existe toujours un chemin alternatif entre deux identifiants de classes. Fusionner de manière gloutonne les classes est alors cohérent tandis que le nombre de *composantes fortement connexes*

est le même que celui de composantes connexes (voyant le graphe non-orienté).

Cela a pour conséquence de restreindre la capacité de l'algorithme à fusionner rapidement des classes lors de la première passe dans le but de réduire le temps global de traitement. Cependant, si on considère le cas où le graphe est réduit à une chaîne, la construction résultante de la première passe sera alors un *couplage parfait* puisque chaque identifiant aura été fusionné au moins une fois *sans nécessité* de fermeture transitive.

Algorithme 6.6.5 Calcul stochatisque de $\varepsilon(P \vee Q)$

```
1: θ ← {(x, x) | x ∈ π2(M)}
2: passe ← 1                                                    ▷ Init step
3: while passe = 1 ou (passe = 2 et θ ≠ ∅) faire
4:     pour tout (x, classe(x)) ∈ θ faire
5:         si passe = 2 alors
6:             classe(x) ← x
7:         fin si
8:         pour tout u | ∃t (M(t, x), N(t, u)) faire
9:             si x < y alors
10:                si classe(x) < classe(y) alors
11:                    si classe(y) traité ou passe = 1 alors
12:                        y ← classe(x)
13:                        θ ← θ\{x}
14:                    fin si
15:                sinon si classe(y) < classe(x) alors
16:                    si classe(x) traité ou x = classe(x) ou passe = 1 alors
17:                        x ← classe(y)
18:                        θ ← θ\{x}
19:                    fin si
20:                fin si
21:            sinon si x > y alors
22:                si classe(y) < classe(x) alors
23:                    si classe(x) traité ou x = classe(x) ou passe = 1 alors
24:                        x ← classe(y)
25:                        θ ← θ\{x}
26:                    fin si
27:                fin si
28:            fin si
29:        fin pour
30:    fin pour
31:    passe ← passe + 1
32: fin while
```

	20000	50000	10000
minimum	90	230	480
maximum	220	480	1270
moyen	139	291	640
médian	110	250	510

FIGURE 6.23 : Temps de chargement des partitions (ms) regroupés par taille des jeux de données

6.7 Expériences

Dans cette section, nous présentons nos résultats expérimentaux pour les différentes versions du calcul de $\varepsilon(P \vee Q)$. Nous avons également adapté les requêtes relationnelles proposées dans la section précédente pour les opérateurs $(\wedge)$ et $(-)$.

La principale conclusion à en tirer est que nos considérations exposées dans les sections précédentes se confirment. En effet, nos algorithmes atteignent des temps de traitement décents en regard de la taille plus importante des jeux utilisés et, en particulier, la version de l'algorithme avec passerelle procure de loin les meilleurs performances, quel que soit le jeu de données.

Le protocole en lui-même suit le même plan que celui de la section (6.5) et dans les mêmes conditions, à l'exception des cardinalités. Étant données deux partitions, nous allons donc évaluer les performances des trois algorithmes précédents simulant $P \vee Q$ et décrits par les dénominations suivantes :

- *Union-Find* : c'est la version simplifiée de l'algorithme 6.6.2 sans utiliser de passerelle ;
- *Bridge Union-Find* : mise en œuvre de 6.6.2 ;
- *Randomized Union-Find* : mise en œuvre de 6.6.5 ;

6.7.1 Résultats et analyses

Les résultats sont reportées pour des partitions P et Q définis pour $n = 20\,000$, $50\,000$ and $100\,000$ objets – 35 jeux de données différents pour chaque valeur de n. Nous avons observé dans nos expérimentations qu'il est délicat de gérer des plus grands ensemble d'objets, étant donné que le temps de traitement augmente d'un certain ordre de grandeur et chaque opération binaire de nécessiter une minute de traitement. Le nombre de classes engendrées dans les partitions est compris entre 10 et 40. La distribution des cardinalités des classes est la même que celle observée dans nos précédentes expériences (*c.f.* section 6.5.3).

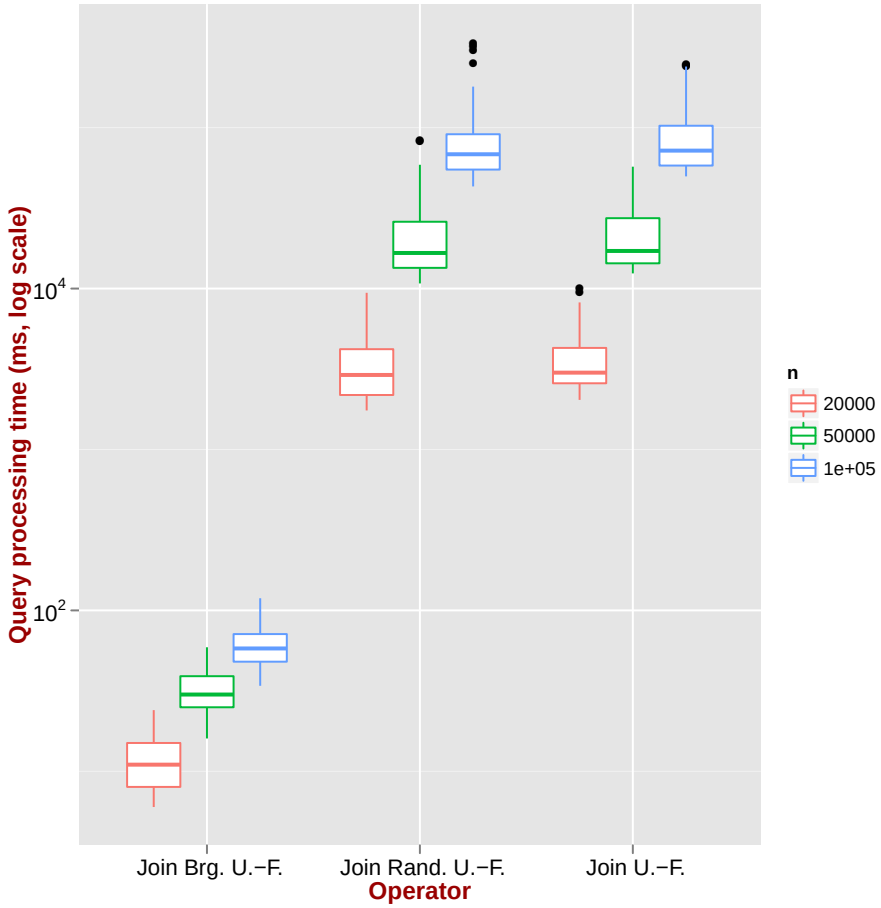


FIGURE 6.24 : Temps de traitement des $\varepsilon(P \vee Q)$ par les 3 algorithmes, mesuré pour $n = 20\,000, 50\,000, 100\,000$

Les résultats de la figure (6.24) représente nos expériences pour l'évaluation de la borne supérieure $(\vee)$. Tandis que les résultats des autres opérateurs sont eux dépeints sur les figures (6.25) et (6.26).

Par ailleurs, les coûts mesurés ne font référence qu'au temps effectivement passé par chaque algorithme. Les temps de chargement des partitions en mémoire sont eux isolés dans la figure (6.23).

Typiquement, le temps de calcul nécessaire pour effectuer $\varepsilon(P \vee Q)$ croît suivant la cardinalité des jeux utilisés : les différents algorithmes appliqués sur des partitions contenant $100\,000$ objets sont dix fois plus lents que leur application sur des partitions n'en contenant que $20\,000$.

La version de l'algorithme (*Bridge Union-Find*) fait exception puisqu'elle expose des résultats convaincants et semble justifiée dans un contexte opérationnel. En effet, en comparaison des autres versions, elle les surpasse toutes de 2 à 3 ordres de grandeurs, sur l'ensemble des jeux de données. Il est intéressant de noter que la version stochastique fournit des résultats étonnamment meilleurs que la version classique sur l'en-

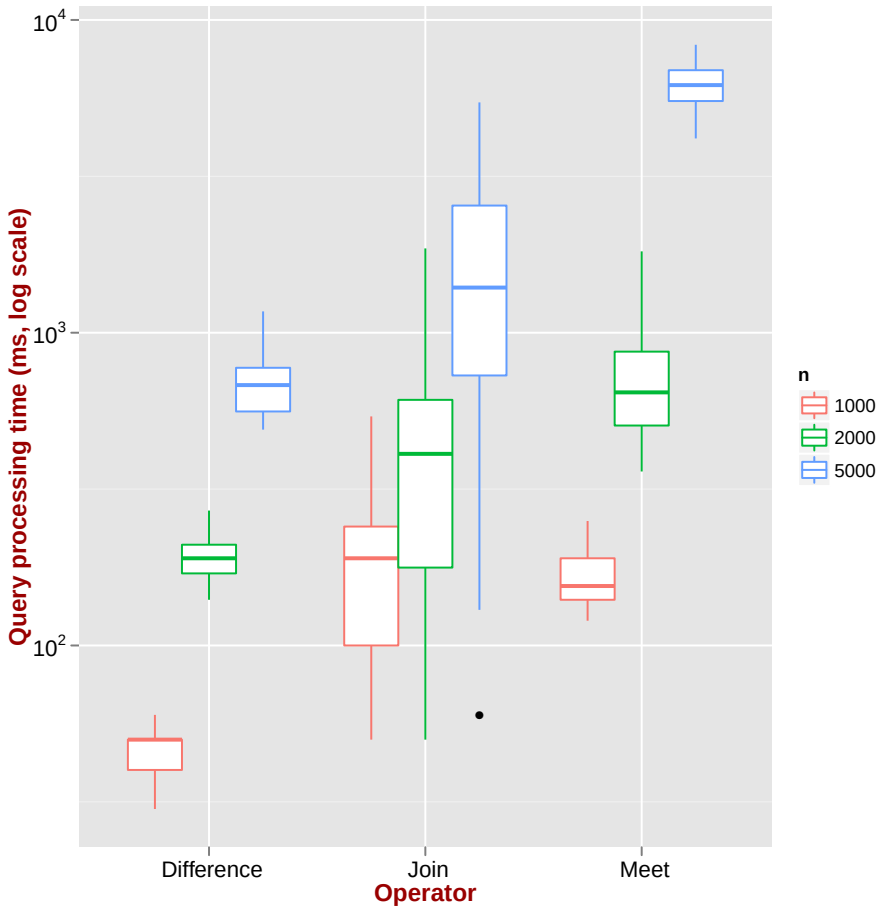


FIGURE 6.25 : Temps de traitement des opérateurs en C++ pour $n = 1\,000, 2\,000, 5\,000$ (pour les opérateurs principaux $-$,$\vee$,$\wedge$, de gauche à droite).

semble des jeux de données. Cela étant dû au fait que la complexité du parcours des classes de chaque opérande $O(n^2)$ domine celle de la fusion des classes qui ne travaille que sur une faible fraction du domaine actif.

6.8 Conclusion

À notre connaissance, il n'existe pas à ce jour de travaux ayant étudiés comment réaliser le traitement de requêtes sur des partitions d'ensembles *génériques*. En particulier, il n'existe pas d'évaluation sur la démarche à adopter pour mettre en œuvre des opérations sur des partitions.

Dans ce chapitre, nous avons proposé une contribution soutenant l'encodage relationnel des partitions par la réalisation de la relation d'appartenance objet-classe, de manière à pouvoir les manipuler et les combiner entre elles. Ce besoin est largement justifié par nos deux précédents chapitres (4) et (5) dans lequel nous exposons un usage

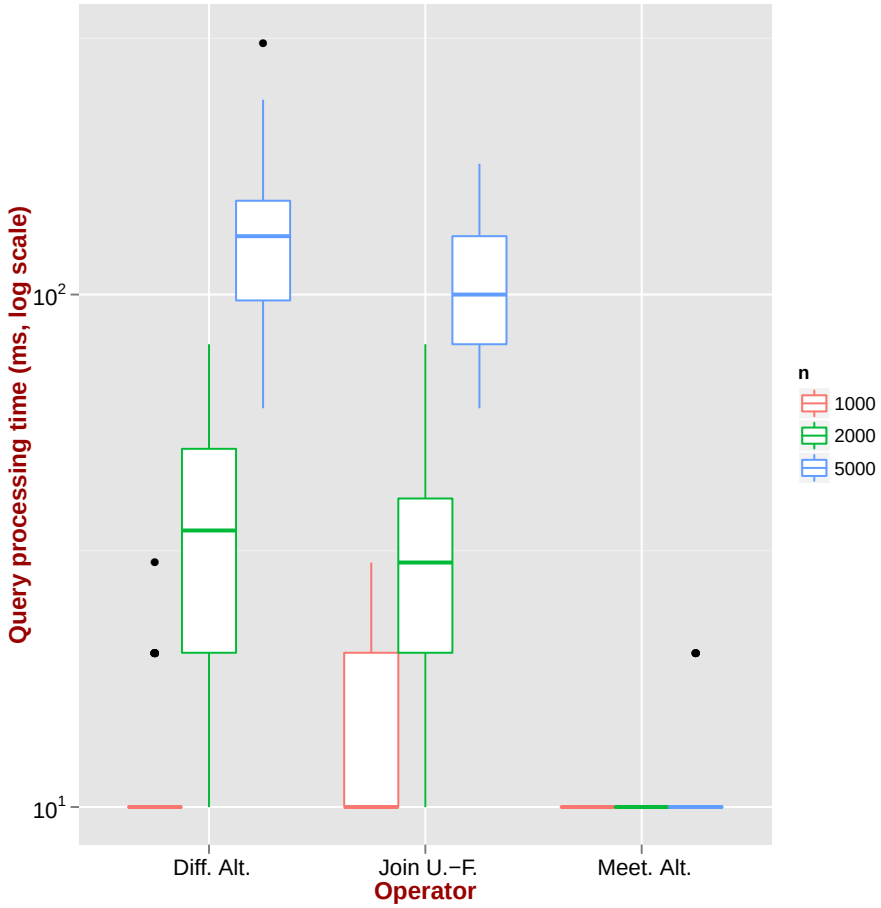


FIGURE 6.26 : Temps de traitement des opérateurs en C++ pour $n = 1\,000, 2\,000, 5\,000$ (pour les versions optimisées des opérateurs $-$, $\vee$ and $\wedge$, de gauche à droite).

concret des partitions.

Nous avons également étudié le cadre algorithmique et avons transcrit chaque opérateur latticiel, opérant sur les partitions, et ensembliste, opérant sur leurs classes, lorsque cela était possible.

Nous avons proposé quelques pistes permettant l'amélioration du comportement de certains opérateurs à travers le stockage d'informations additionnelles.

Les diverses expérimentations que nous avons menées montrent que le calcul des partitions est globalement insoluble à cause de ses maigres performances lorsque la cardinalité de l'univers augmente, même en considérant les versions optimisées des opérations.

Finalement, nous avons proposé une mise en œuvre originale de l'opérateur de $(\vee)$ en mémoire centrale, surpassant le moteur SQL par la réduction du nombre d'E/S et l'incapacité naturelle de SQL à contourner le problème du calcul de la fermeture transitive.

Nous revendiquons donc qu'un moteur de requêtes opérant sur des partitions d'ensembles, qui soit capable de travailler sur des univers différents, devrait être conçu dans le but de pleinement exploiter le potentiel d'optimisations, notamment en suivant les propriétés algébriques des partitions.

7

Conclusions et Perspectives

Cette thèse a été motivée dans un premier temps dans l'intention d'établir des applications modélisées par le biais de partitions d'ensemble. En particulier, l'augmentation régulière des applications du clustering et l'émergence des données ouvertes, mène à un partage grandissant de ces données et, éventuellement, de leurs résultats d'analyse. D'où la nécessité de gérer les résumés qui en découlent, et ce, en fonction du contexte où ils apparaissent. Cela nous a donc mené à l'étude de certaines structures algébriques les mettant en relation.

Cette préoccupation nous a donc amené à l'étude générale des treillis et à leurs propriétés algébriques. Celles-ci nous ont permis de construire des produits de treillis $\Pi_\Omega \times L$ et $\mathcal{P}(\Omega) \times L$ permettant de contextualiser les résumés naissant de l'application d'algorithmes de clustering et ceux de requêtes de l'algèbre relationnelle, la dernière octroyant une représentation de chaque classe d'une partition d'ensemble.

Dans le chapitre 4, nous avons exposé la possibilité d'exprimer de manière succinte certaines dépendances apparaissant au sein des données à l'aide des partitions annotées dans $\Pi^\Omega_\mathcal{A}$ et permettant de manipuler les modèles associées à certaines requêtes de l'algèbre relationnelle. En particulier, nous avons montré la possibilité de construire des classes d'équivalence d'attributs $\mathcal{X} \in \mathcal{P}(\mathcal{P}(\mathcal{A}))$.

De plus, puisque la structure associée est toujours maximalement bornée, cela ouvre la voie au maintien de certains modèles par opposition à ceux qui en dépendent. La construction des résultats de requêtes est alors réalisable par induction sur les éléments de $\mathcal{A}$ et la recherche de la classe correspondante.

Néanmoins, la construction de ces classes ne représente pas nécessairement la pa-

nacée : la contrepartie associée au représentant de chaque classe est celle des parties génératrices de celles-ci, qui permettrait de dissocier la construction d'une classe particulière et de déduire les parties génératrices d'une autre sur le restant des attributs. Ceci figure donc un axe d'amélioration important de notre modélisation.

Par ailleurs, nous avons considéré la construction des modèles qui correspondent effectivement au résultat d'une requête, soit les parties de Ω représentant des projections sur les attributs dissociant les tuples n'ayant pas la même valeur. Ceux-ci nous assurent la construction d'$\sqcap$-expression engendrant tous les résultats de requêtes.

Les expressions duales selon $(\sqcup)$ ne sont définies que pour les représentants de classes, étant données qu'elles symbolisent des dépendances multivaluées. Ceci implique donc en règle générale : $P_{X\cup Y} \sqcup P_{X\cup Z} \leq P_X$. Il n'est donc pas possible de recouvrer certaines relations par ce biais. Néanmoins, il existe une alternative.

Par exemple, la fonction $f : \mathcal{P}(\mathcal{A}) \to \Pi^\Omega$ induit l'équivalence suivante :

$$t, u \in [.]_X \iff \forall A \in X, t[A] = u[A] \tag{7.1}$$

Or, il est également possible de maintenir uniquement, par exemple, les tuples qui sont égaux sur au moins une projection canonique, soit en remplaçant $(\forall)$ par le quantificateur existentiel $(\exists)$. On obtiendrait alors le couple d'identités suivants :

$$P^\forall_{X\cup Y} = P^\forall_X \sqcap P^\forall_Y \tag{7.2}$$

$$P^\forall_X = P^\forall_{X\cup Y} \sqcup P^\forall_{X\cup Z} \sqcup P^\exists_{X\cup Y\cup Z} \tag{7.3}$$

Ceci permettrait la navigabilité de la structure sans la moindre contrainte et ouvrirait de nouvelles perspectives dans le cadre de la navigation de cube OLAP. On notera que $P^\exists_{X\cup Y\cup Z}$ ne sera pas nécessairement une partition mais plus vraisemblablement une collection dans $\mathcal{P}(\mathcal{P}(\Omega))$.

Enfin, cette structure, par l'ajout de nouvelles relations sur le treillis $(\mathcal{P}(\mathcal{A}), \subseteq)$, induit nécessairement une extension de l'ordre. La relation naturelle représentant des dépendances fonctionnelles triviales entre les attributs, permet d'énumérer de nouvelles dépendances. Il serait donc intéressant d'utiliser les partitions annotées dans le cadre d'applications exploitant des dépendances sur les tuples[1] afin que celles-ci ne représentent plus les propriétés combinatoires des tuples mais les construisent.

Le chapitre 5 est essentiellement plus théorique. Dans celui-ci, nous avons étudié la possibilité d'adjoindre de nouveaux opérateurs aux treillis des partitions $(\Pi^\Omega, \leq, \wedge, \vee)$ afin de modéliser selon une approche constructive des opérations de complémentations compatibles avec la non-distributivité de la structure. En effet, la dépendance s'exerçant entre les parties élémentaires et décrivant de manière exhaustive les agrégats au sein d'une partition, impose certaines contraintes qui ignorent la séparabilité entre les agrégats définis dans des partitions différentes, ce qui limite l'intérêt d'une telle approche.

La représentation induite des partitions par $\Pi^\Omega \to \mathcal{P}(\mathcal{FI}_m\Pi^\Omega)$ impose donc de modéliser des expressions ensemblistes et donc de renvoyer des résultats de la même nature.

[1] tuple-generating dependencies

Une piste de travail permettant de consolider cette approche serait de généraliser le modèle de sorte à considérer directement des intervalles de la structure.

En effet, quoi qu'il en soit, l'intervalle symbolisant les bornes inférieures et supérieures contenant les éléments médians demeure un ensemble potentiellement très grand pour être exploitable puisqu'aucune méthode ou critère n'existe pour choisir un élément dans la structure. Il serait intéressant de chercher des méthodes itératives permettant de converger vers des sous-ensembles de partitions incluses dans cet intervalle.

Une autre piste de recherche envisagée est de considérer une modification du modèle. En effet, ici, chaque partition est définie sur l'ensemble Ω et ne dispose d'aucune espèce de structure. Il serait alors intéressant d'envisager des valuations concrètes selon $\mathbb{R}^+$, étiquetant les relations entre chaque point dans Ω. En particulier, l'absence d'axiomatisation sur la construction des résultats de clustering ne permet pas de guider clairement la combinatoire de ces résultats et de proposer une ou plusieurs expressions fiables dans la recherche d'un compromis.

Enfin dans le dernier chapitre, nous avons proposé une modélisation conceptuelle des partitions d'ensemble en l'absence d'un contexte particulier dans le but d'assurer le support de leur calcul dans un environnement relationnel comme un SGBD. À cet égard, nous avons proposé une modélisation relationnelle des opérateurs latticiels et des opérateurs ensemblistes, et le schéma fonctionnel permettant de transcrire et simuler les opérations algébriques dans ce système.

Ayant constaté de faibles performances, en particulier, pour l'opérateur de $(\vee)$, nous avons proposé une mise en œuvre dédiée du stockage des partitions et de l'implémentation des opérateurs en dehors des SGBD relationnels, bien qu'ils soient suffisamment expressifs pour gérer et manipuler de telles relations.

Typiquement, la simplicité de notre modèle nous a permis de développer en dehors des contraintes opérationnelles inhérentes aux SGBDs, des algorithmes basés sur l'emploi de procédures de type *Union-Find* pour la mise en œuvre de l'opérateur de la borne supérieure $(\vee)$. Cette technique efficace est inconcevable en pratique au sein des moteurs de requêtes relationels et requiert le développement de d'un moteur *ad hoc* d'évaluation de propositions sur les partitions.

En sus, nous avons proposé la modélisation formelle de la combinatoire du système de représentant des classes d'une partition par la structure $\mathcal{P}_{\min}(\Omega)$. Néanmoins, nous avons montré que lorsqu'il s'agit de minimiser le nombre de parcours au sein des classes associées à chaque opérande, il était nécessaire d'intégrer des contraintes supplémentaires liées à la structure naissante du graphe biparti induit par l'intersection de leurs classes. Un axe de recherche intéressant serait de caractériser formellement les contraintes de fonctionnement de nos algorithmes en fonction d'une structure $\mathcal{P}_{\min}(\Omega) \times I$ – représentant les interactions entre les classes à fusionner et les intersections disponibles pour les réaliser – et obtenir une définition inductive d'un algorithme commun assurant la réalisation du point fixe.

Dans un contexte opérationnel, où on aurait à gérer un ensemble de partitions, il

serait intéressant d'ajouter à notre représentation $\varepsilon : \mathcal{P}(\Omega) \rightarrow \Omega$ qui associe à chaque classe d'une partition, son représentant, la construction inverse $\varepsilon^c : \Omega \times \mathbb{U} \rightarrow \mathcal{P}(\Omega)$ permettant la recherche des classes en fonction de leur représentant et d'une ou plusieurs informations contextuelles, propres aux données concrètes $\mathbb{U}$. Cela ouvrirait la voie à un mécanisme d'indexation des classes permettant d'assouplir les conditions d'accès à chaque classe, sans avoir à charger l'ensemble de la partition.

Plus généralement, la solution qui nous semble la plus appropriée, au moins dans un premier temps, pour faciliter l'opérationnalisation d'une base de données de partitions est d'employer un support de stockage capable de charger et filtrer efficacement les représentations arborescentes des partitions puis d'y adjoindre un moteur d'évaluation relié aux implémentations de nos opérateurs.

8

Annexe

8.1 Annexe A : Requêtes SQL

Dans cette annexe, nous mettons à disposition les requêtes SQL pour chaque opérateur parmi l'ensemble $\{-, \wedge, \vee\}$, présenté dans le chapitre 5. Pour chaque requête, on part du principe qu'il existe deux tables M et N matérialisant l'encodage relationnelle des partitions P et Q

Nous rappelons que les schémas de relation sont $M(\mathsf{elt} : \Omega, \mathsf{block} : \Omega)$ and $N(\mathsf{elt} : \Omega, \mathsf{block} : \Omega)$. Les identifiants des éléments sont (a) uniques au sein de chaque table, (b) sont partagées parmi toutes les tables M et N, et (c) doivent être totalement ordonnés.

Sans perte de généralité, les colonnes elt sont déclarées de type *entier positif* pour appliquer les contraintes.

```
SELECT outM.elt,
       outN.block
FROM $1 outM JOIN $2 outN USING (elt)
WHERE EXISTS (
    ( SELECT *
      FROM $1 inM JOIN $2 inM USING (elt)
      WHERE outM.block=inM.block AND
            outN.block<>inN.block
    )
  UNION
    ( SELECT *
      FROM $1 inM2 JOIN $2 inN2 USING (elt)
    WHERE outM.block<>inM2.block AND
```

```
            outM.block=inN2.block
    )
              );
```

Listing 8.1 : Version initiale de l'opérateur de différence $(-)$

```
SELECT outM2.elt,
       foo.min AS block
FROM
( SELECT block,
         min(elt),
         count(elt)
  FROM $1 outM GROUP BY block
) AS foo
JOIN
( SELECT block,
         min(elt),
         count(elt)
  FROM $2 outN GROUP BY block
) AS bar
USING (min, count)
JOIN $1 outM2 ON (outM2.block=foo.block)
WHERE EXISTS (
    ( SELECT *
      FROM $1 inM JOIN $2 inN USING (elt)
      WHERE foo.block=inM.block AND
            bar.block<>inN.block
    )
    UNION
    ( SELECT *
      FROM $1 inM2 JOIN $2 inN2 USING (elt)
      WHERE foo.block<>inM2.block AND
            bar.block=inN2.block
    )
             );
```

Listing 8.2 : Version optimisée de l'opérateur de différence $(-)$

```
SELECT foo.elt AS elt,
       bar.elt AS block
FROM
( SELECT elt,
       outM.block AS block1,
       outN.block AS block2
  FROM $1 outM JOIN $2 outN USING (elt)
) AS foo
JOIN
( SELECT elt,
       outM2.block AS block1,
       outN2.block AS block2
  FROM $1 outM2 JOIN $2 outN2 USING (elt)
) AS bar
ON (foo.block1=bar.block1 AND
  foo.block2=bar.block2)
```

```
WHERE NOT EXISTS (
  SELECT *
  FROM $1 inM JOIN $2 inN USING (elt)
  WHERE inM.block=foo.block1 AND
        inN.block=foo.block2 AND
        elt<bar.elt
  );
```

Listing 8.3 : Version initiale de l'opérateur $(\wedge)$

```
SELECT outM.elt,
       min(elt) OVER (
           PARTITION BY (outM.block,
                         outN.block)
                        )
FROM $1 outM JOIN $2 outN USING (elt);
```

Listing 8.4 : Version fenêtrée de $(\wedge)$

```
WITH RECURSIVE
BTC(bfrom, bto) AS (
    ( SELECT mb,
             mb
    FROM BJoin
  )
    UNION
    ( SELECT t.bfrom,
             j2.mb
    FROM BTC t
      JOIN BJoin j1 ON (t.bto=j1.mb)
      JOIN BJoin j2 ON (j1.nb=j2.nb)
    )
             ),
BJoin (mb, nb) AS (
    SELECT inM.block,
           inN.block
    FROM $1 inM JOIN $2 inN USING (elt)
                )
SELECT outM.elt,
       t.bfrom AS block
FROM $1 outM JOIN BTC t ON (outM.block=t.bto)
WHERE NOT EXISTS (
    SELECT *
    FROM $1 inM2 JOIN BTC t2 ON (inM2.block=t2.bto)
    WHERE inM2.elt=outN.elt AND
          t2.bfrom < t.bfrom
                );
```

Listing 8.5 : Opérateur $(\vee)$

8.2 Annexe B : Plans d'exécution

Nous mettons également à disposition quelques plans d'exécution bruts pour les opérateurs $\{-, \wedge, \vee\}$. Ils illustrent la complexité inhérente des différentes étapes nécessaires à la mise en œuvre de leur calcul. Chaque terme (c.-à-d. les représentations relationnelles des partitions), qui sont les paramètres des requêtes, sont définies sur un univers de 5000 objets tandis que la seconde partition est obtenue depuis la première, suivant le protocole exposée dans la section (6.5.2).

Temps d'exécutions	Nombre de lignes	Nœud
11119	3552	Nested Loop Anti Join | Join Filter : (((outM.block = inM.block) AND (outN.block <> inN.block)) OR ((outM.block <> inM2.block) AND (outM.block = inM2.block)))
10	5000	Merge Join | Cond : (outM.id = outN.id)
3.2	5000	Index Scan using $1_elt_key on $1 outM
6	5000	Index Scan using $2_elt_key on $2 outN
7535	3740	Materialize
10	5000	Hash Join | Cond : (inM.elt = inN.elt)
4.7	5000	Seq Scan on $1 inM
4.8	5000	Hash Buckets : 1024 Memory Usage : 137kB
4.9	5000	Seq Scan on $2 inN

TABLE 8.1 : Plan d'exécution de la requête pour l'opérateur $(-)$

Temps d'exécutions	Nombre de lignes	Nœud
80068	5000	Hash Anti Join | Cond : ((outM.block = outM2.block) AND (outN.block = outN2.block)) | Join Filter : (outM.elt < outM2.elt)
1367	1044784	Merge Join | Cond : ((outM.block = outM2.block) AND (outN.block = outN2.block)) | Join Filter : (outM.elt >= outM2.elt)
7.3	5000	Sort | Key : outM.block, outN.block | Method : quicksort Memory : 324kB
9.8	5000	Hash Join | Cond : (outM.elt = outN.elt)
4.1	5000	Seq Scan on $1 outM
4.9	5000	Hash Buckets : 1024 Memory Usage : 137kB
4.2	5000	Seq Scan on $2 outN
691	2084556	Sort | Key : outM.block, outN.block | Method : quicksort Memory : 324kB
9.7	5000	Hash Join | Cond : (outM.elt = outN.elt)
3.9	5000	Seq Scan on $1 outM
4.6	5000	Hash Buckets : 1024 Memory Usage : 137kB
4	5000	Seq Scan on $2 outN
5.2	5000	Hash Buckets : 1024 Memory Usage : 157kB
9.8	5000	Hash Join | Cond : (outM.elt = outN.elt)
4	5000	Seq Scan on $1 outM
4.6	5000	Hash Buckets : 1024 Memory Usage : 137kB
3.9	5000	Seq Scan on $2 outN

TABLE 8.2 : Plan d'exécution de la requête pour l'opérateur $(\wedge)$

Temps d'exécutions	Nombre de lignes	Nœud
1488	5000	Hash Anti Join | Cond : (p1.elt = p2.elt) | Join Filter : (t2.n < t.n)
		CTE cjoin
11.3	5000	Hash Join | Cond : (p1.elt = p2.elt)
4.5	5000	Seq Scan on $1 p1
6.2	5000	Hash Buckets : 1024 Memory Usage : 137kB
5.5	5000	Seq Scan on $2 p2
		CTE t
2857	32	Recursive Union
39	5000	CTE Scan on cjoin
6754	1290969	Merge Join | Cond : (t.p = j1.n)
0.1	8	Sort | Key : t.p | Method : quicksort Memory : 17kB
0.02	8	WorkTable Scan on t
9141	2463782	Materialize
12281	2338475	Sort | Key : j1.n | Method : external merge Disk : 41128kB
8498	2339340	Merge Join | Cond : (j1.p = j2.p)
22	5000	Sort | Key : j1.p | Method : quicksort Memory : 324kB
11.2	5000	CTE Scan on cjoin j1
4003	2339328	Sort | Key : j2.p | Method : quicksort Memory : 324kB
4.35	5000	CTE Scan on cjoin j2
21.2	9344	Merge Join | Cond : (p1.block = t.p)
12.4	5000	Sort | Key : p1.block | Method : quicksort Memory : 324kB
4.2	5000	Seq Scan on $1 p1
6.7	9332	Materialize
0.07	32	Sort | Key : t.p | Method : quicksort Memory : 18kB
43612	32	CTE Scan on t
129	9344	Hash Buckets : 4096 Batches : 262144 Memory Usage : 1kB
12	9344	Merge Join | Cond : (p2.block = t2.p)
4.9	5000	Sort | Key : p2.block | Method : quicksort Memory : 324kB
2.2	5000	Seq Scan on $1 p2
3.9	9332	Materialize
0.04	32	Sort | Key : t2.p | Method : quicksort Memory : 18kB
0.02	32	CTE Scan on t t2

TABLE 8.3 : Plan d'exécution de la requête pour l'opérateur $(\vee)$

Liste des tableaux

Table des figures

Bibliographie

[1] *La philosophie de l'atomisme logique*. 10

[2] *Philosophical Magazine*. Taylor & Francis., 1854. 20

[3] Daniel J Abadi, Adam Marcus, Samuel R Madden, and Kate Hollenbach. Scalable semantic web data management using vertical partitioning. In *Proceedings of the 33rd international conference on Very large data bases*, pages 411–422. VLDB Endowment, 2007. 103

[4] Sanjay Agrawal, Vivek Narasayya, and Beverly Yang. Integrating vertical and horizontal partitioning into automated physical database design. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 359–370. ACM, 2004. 80

[5] Michael Akinde, Damianos Chatziantoniou, Theodore Johnson, and Samuel Kim. The MD-join : An operator for complex OLAP. In *Proc. ICDE*, pages 524–533, 2001. 150

[6] Sihem Amer-Yahia, Fang Du, and Juliana Freire. A comprehensive solution to the xml-to-relational mapping problem. In *Proc. of the 6th ACM Int. Workshop on Web Information and Data Management (WIDM'04)*, 2004. 150

[7] Ofer Arieli and Arnon Avron. The logical role of the four-valued bilattice. In *Proc. LICS'98*, pages 218–226. IEEE Press, 1998. 141

[8] Kenneth J. Arrow. *Social Choice and Individual Values*. John Wiley and Sons, New York, NY, 1951. 110

[9] Nurzhan Bakibayev, Dan Olteanu, and Jakub Závodnỳ. Fdb : A query engine for factorised relational databases. *Proceedings of the VLDB Endowment*, 5(11) :1232–1243, 2012. 80

[10] Andrey Balmin and Yannis Papakonstantinou. Storing and querying xml data using denormalized relational databases. *The VLDB Journal - The International Journal on Very Large Data Bases*, 14(1) :30–49, 2005. 103

[11] Bernhard Banaschewski and Marcel Erné. On krull's separation lemma. *Order*, 10(3) :253–260, 1993. 71

[12] M. Barbut. Médiane, distributivité, éloignements. *Mathématiques et Sciences Humaines*, 70 :5–31, 1980. 112

[13] Andrew Barron, Jorma Rissanen, and Bin Yu. The minimum description length principle in coding and modeling. *Information Theory, IEEE Transactions on*, 44(6) :2743–2760, 1998. 10

[14] J.-P. Barthelemy and B. Leclerc. The median procedure for partition. *AMS DIMACS Series in Discrete Math.*, 19 :3–34, 1995. 14, 125

[15] Nuel D Belnap Jr. A useful four-valued logic. In *Modern uses of multiple-valued logic*, pages 5–37. Springer, 1977. 141

[16] Jean-Yves Béziau. Logic may be simple. logic, congruence and algebra. *Logic and Logical Philosophy*, 5 :129–147, 2003. 35

[17] G. Birkhoff and S.A. Kiss. A ternary operation in distributive lattices. *Bull. Amer. Math. Soc.*, 53 :749–752, 1947. 14, 111

[18] Garrett Birkhoff and Orrin Frink. Representations of lattices by sets. *Transactions of the American Mathematical Society*, pages 299–316, 1948. 69

[19] Philip Bohannon, Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. Conditional functional dependencies for data cleaning. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 746–755. IEEE, 2007. 80

[20] P. A. Boncz, T. Neumann, and O Erling. TPC-H Analyzed : Hidden Messages And Lessons Learned From An Influential Benchmark. In M. Poess and R. Niambar, editors, *Proceedings of the TPC Technology Conference on Performance Evaluation & Benchmarking (TPCTC, 2013)*, pages –, August 2013. 99, 104

[21] George Boole. *The mathematical analysis of logic*. Philosophical Library, 1847. 11

[22] George Boole. *An investigation of the laws of thought : on which are founded the mathematical theories of logic and probabilities*. Dover Publications, 1854. 11

[23] Hanen Brahmi and Sadok Ben Yahia. Constrained closed non derivable data cubes. In Shuigeng Zhou, Songmao Zhang, and George Karypis, editors, *Advanced Data Mining and Applications*, volume 7713 of *Lecture Notes in Computer Science*, pages 766–778. Springer Berlin Heidelberg, 2012. 79

[24] Ronald L Breiger, Scott A Boorman, and Phipps Arabie. An algorithm for clustering relational data with applications to social network analysis and comparison with multidimensional scaling. *Journal of mathematical psychology*, 12(3) :328–383, 1975. 9

[25] Christopher Brewster, Kieron O'Hara, Steve Fuller, Yorick Wilks, Enrico Franconi, Mark A Musen, Jeremy Ellman, and Simon Buckingham Shum. Knowledge representation with ontologies : the present and future. *IEEE Intelligent Systems*, pages 72–81, 2004. 9

[26] M.J. Carey, J. Kienan, J. Shanugasundaram, E. Shekita, and S. Subramanian. Xperanto : Middleware for publishing object-relational data as xml documents. In *Proc. of the 26th Int. Conf. on Very Large Databases (VLDB'2000)*, 2000. 150

[27] Alain Casali, Sebastien Nedjar, Rosine Cicchetti, and Lotfi Lakhal. Closed cube lattices. In *New Trends in Data Warehousing and Data Analysis*, pages 1–20. Springer, 2009. 79

[28] Lena Chang and James F Korsh. Canonical coin changing and greedy solutions. *Journal of the ACM (JACM)*, 23(3) :418–422, 1976. 21

[29] Yann Chevaleyre, Ulle Endriss, Jérôme Lang, and Nicolas Maudet. Preference handling in combinatorial domains : From ai to social choice. *AI Magazine*, 29(4) :37, 2009. 110

[30] Fei Chiang and Renée J Miller. Discovering data quality rules. *Proceedings of the VLDB Endowment*, 1(1) :1166–1177, 2008. 80

[31] Benny Chor and Ronald L Rivest. A knapsack-type public key cryptosystem based on arithmetic in finite fields. *Information Theory, IEEE Transactions on*, 34(5) :901–909, 1988. 21

[32] Aaron Clauset, Cosma Rohilla Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *SIAM Review*, 51(4) :661–703, November 2009. 166

[33] Luther Elic Cleborn. Dedekind domains and rings of quotient. 15(1) :59–64, 1965. 20

[34] Nino Cocchiarella. Logical atomism, nominalism, and modal logic. *Synthese*, 31(1) :23–62, 1975. 9

[35] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13 :377–387, June 1970. 148

[36] Sylvain Conchon and Jean-Christophe Filliâtre. A persistent union-find data structure. In *Proceedings of the 2007 workshop on Workshop on ML*, pages 37–46. ACM, 2007. 187

[37] Graham Cormode, Lukasz Golab, Korn Flip, Andrew McGregor, Divesh Srivastava, and Xi Zhang. Estimating the confidence of conditional functional dependencies. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 469–482. ACM, 2009. 80

[38] Alan Day. Congruence normality : the characterization of the doubling class of convex sets. *Algebra Universalis*, 31(3) :397–406, 1994. 118

[39] Edsger Wybe Dijkstra. *A discipline of programming*. 1976. 183

[40] Robert P Dilworth. Lattices with unique irreducible decompositions. In *The Dilworth Theorems*, pages 93–99. Springer, 1990. 116

[41] RP Dilworth and Peter Crawley. Decomposition theory for lattices without chain conditions. *Transactions of the American Mathematical Society*, pages 1–22, 1960. 69

[42] Guozhu Dong, Leonid Libkin, and Limsoon Wong. Local properties of query languages. *Theor. Comput. Sci.*, 239 :277–308, May 2000. 160

[43] F. Dumonceaux, G. Raschia, and M. Gelgon. An algebraic approach to ensemble clustering. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pages 1301–1306, Aug 2014. 14, 115

[44] Frédéric Dumonceaux, Guillaume Raschia, and Marc Gelgon. Materializing data cubes as set partitions of tuples. 75

[45] Frédéric Dumonceaux, Guillaume Raschia, and Marc Gelgon. A first attempt to computing generic set partitions : Delegation to an sql query engine. In Hendrik Decker, Lenka LhotskÃi, Sebastian Link, Marcus Spies, and RolandR. Wagner, editors, *Database and Expert Systems Applications*, volume 8645 of *Lecture Notes in Computer Science*, pages 433–440. Springer International Publishing, 2014. 13, 15, 99, 147

[46] Frédéric Dumonceaux, Guillaume Raschia, and Marc Gelgon. Computing partitions within sql queries : A dead end. Technical report, January 2013. 15

[47] J Michael Dunn and Gary Hardegree. *Algebraic methods in philosophical logic*. Oxford University Press, 2001. 57

[48] Wojciech Dzik, Ewa Orłowska, and Clint van Alten. Relational representation theorems for lattices with negations : A survey. In *Theory and Applications of Relational Structures as Knowledge Instruments II*, pages 245–266. Springer, 2006. 145

[49] H.M. Edwards. *Galois Theory*. Graduate Texts in Mathematics. Springer New York, 1997. 20

[50] Ronald Fagin. Multivalued dependencies and a new normal form for relational databases. *ACM Transactions on Database Systems (TODS)*, 2(3) :262–278, 1977. 96

[51] Wenfei Fan, Jianzhong Li, Nan Tang, and Wenyuan Yu. Incremental detection of inconsistencies in distributed data. *Knowledge and Data Engineering, IEEE Transactions on*, 26(6) :1367–1383, 2014. 103

[52] L. M. G. Feijs and R. C. van Ommering. Relation partition algebra —mathematical aspects of uses and part-of relations. *Science of Computer Programming*, 33(2) :163–212, 2 1999. 150

[53] Xiaoli Zhang Fern and Carla E Brodley. Random projection for high dimensional data clustering : A cluster ensemble approach. In *ICML*, volume 3, pages 186–193, 2003. 12

[54] Josep Maria Font, Ramon Jansana, and Don Pigozzi. A survey of abstract algebraic logic. *Studia Logica*, 74(1-2) :13–97, 2003. 36

[55] Michael Fredman and Michael Saks. The cell probe complexity of dynamic data structures. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 345–354. ACM, 1989. 175

[56] Zvi Galil and Giuseppe F. Italiano. Data structures and algorithms for disjoint set union problems. *ACM Comput. Surv.*, 23(3) :319–344, September 1991. 175

[57] Eve Garnaud, Sofian Maabout, and Mohamed Mosbah. Functional dependencies are helpful for partial materialization of data cubes. *Annals of Mathematics and Artificial Intelligence*, 73(1-2) :245–274, 2015. 80, 84

[58] Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. Producing power-law distributions and damping word frequencies with two-stage language models. *J. Mach. Learn. Res.*, 12 :2335–2382, July 2011. 166

[59] Matteo Golfarelli and Stefano Rizzi. A methodological framework for data warehouse design. In *Proceedings of the 1st ACM international workshop on Data warehousing and OLAP*, pages 3–9. ACM, 1998. 80

[60] Jim Gray, Surajit Chaudhuri, Adam Bosworth, Andrew Layman, Don Reichart, Murali Venkatrao, Frank Pellow, and Hamid Pirahesh. Data cube : A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data Mining and Knowledge Discovery*, 1(1) :29–53, 1997. 84

[61] Nicola Guarino. Formal ontology, conceptual analysis and knowledge representation. *International journal of human-computer studies*, 43(5) :625–640, 1995. 9

[62] G Guilbaud. Les théories de l'intérêt général et le problème logique de l'agrégation. *Revue économique*, 63 :659–720. 111

[63] Ashish Gupta, Venky Harinarayan, and Dallan Quass. Aggregate-query processing in data warehousing environments. 1995. 79

[64] Terry Halpin and Tony Morgan. *Information Modeling and Relational Databases*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2008. 149

[65] Markus Held and Wolfgang Blochinger. Structured collaborative workflow design. *Future Generation Computer Systems*, 25(6) :638–653, 2009. 12

[66] Edith Hemaspaandra, Lane A Hemaspaandra, and Jörg Rothe. Anyone but him : The complexity of precluding an alternative. *Artificial Intelligence*, 171(5) :255–285, 2007. 110

[67] Samuel S Holland. Distributivity and perspectivity in orthomodular lattices. *Transactions of the American Mathematical Society*, pages 330–343, 1964. 119

[68] Yi Hong, Sam Kwong, Yuchou Chang, and Qingsheng Ren. Unsupervised feature selection using clustering ensembles and population based incremental learning algorithm. *Pattern Recognition*, 41(9) :2742–2756, 2008. 109

[69] John E Hopcroft, Jeffrey David Ullman, and Alfred Vaino Aho. *Data structures and algorithms*. Addison-Wesley, 1983. 175

[70] Xiaohua Hu and Illhoi Yoo. Cluster ensemble and its applications in gene expression analysis. In *Proceedings of the second conference on Asia-Pacific bioinformatics-Volume 29*, pages 297–302. Australian Computer Society, Inc., 2004. 109

[71] Anil K Jain and Richard C Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988. 11

[72] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering : a review. *ACM computing surveys (CSUR)*, 31(3) :264–323, 1999. 11

[73] Alekh Jindal and Jens Dittrich. Relax and let the database do the partitioning online. In Malu Castellanos, Umeshwar Dayal, and Wolfgang Lehner, editors, *Enabling Real-Time Business Intelligence*, volume 126 of *Lecture Notes in Business Information Processing*, pages 65–80. Springer Berlin Heidelberg, 2012. 80

[74] Bjarni Jónsson and Alfred Tarski. Boolean algebras with operators. part i. *American journal of mathematics*, pages 891–939, 1951. 70

[75] Irving Kaplansky. Any orthocomplemented complete modular lattice is a continuous geometry. *Annals of Mathematics*, pages 524–541, 1955. 119

[76] Grigoris Karvounarakis and Todd J. Green. Semiring-annotated data : queries and provenance ? *ACM SIGMOD Record*, 3(41) :5–14, 2012. 150

[77] Wolfgang Keller. Mapping objects to tables : A pattern language. In *Proceedings of EurPLoP*, 1997. 150

[78] Ralph Kimball, Laura Reeves, Warren Thornthwaite, Margy Ross, and Warren Thornwaite. *The Data Warehouse Lifecycle Toolkit : Expert Methods for Designing, Developing and Deploying Data Warehouses with CD Rom*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1998. 84

[79] Dan Klein, Sepandar D Kamvar, and Christopher D Manning. From instance-level constraints to space-level constraints : Making the most of prior knowledge in data clustering. 2002. 109

[80] Jon Kleinberg. An impossibility theorem for clustering. *Advances in neural information processing systems*, pages 463–470, 2003. 110

[81] Laks V. S. Lakshmanan, Jian Pei, and Jiawei Han. Quotient cube : How to summarize the semantics of a data cube. In *Proceedings of the 28th International Conference on Very Large Data Bases*, VLDB '02, pages 778–789. VLDB Endowment, 2002. 78

[82] Bruno Leclerc. Medians and majorities in semimodular lattices. *SIAM Journal on Discrete Mathematics*, 3(2) :266–276, 1990. 114

[83] Bruno Leclerc. Lattice valuations, medians and majorities. *Discrete Mathematics*, 111(1-3) :345 – 356, 1993. 111

[84] Xiaolei Li, Jiawei Han, and Hector Gonzalez. High-dimensional olap : A minimal cubing approach. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, VLDB '04, pages 528–539. VLDB Endowment, 2004. 100

[85] Grzegorz Malinowski. Many-valued logic and its philosophy. *Handbook of the History of Logic*, 8 :13–94, 2007. 141

[86] Andrey V. Malishevski. Path independence in serial–parallel data processing. *Mathematical Social Sciences*, 27(3) :335–367, 1994. 155

[87] George Markowsky. The factorization and representation of lattices. *Transactions of the American Mathematical Society*, 203 :185–200, 1975. 127

[88] George Markowsky. Primes, irreducibles and extremal lattices. *Order*, 9(3) :265–290, 1992. 69, 127

[89] Silvano Martello and Paolo Toth. Optimal and canonical solutions of the change making problem. *European Journal of Operational Research*, 4(5) :322–329, 1980. 21

[90] F.R McMorris, H.M Mulder, and R.C Powers. The median function on distributive semilattices. *Discrete Applied Mathematics*, 127(2) :319 – 324, 2003. Ordinal and Symbolic Data Analysis (OSDA '98), Univ. of Massachusetts, Amherst, Sept. 28-30, 1998. 111

[91] B. G. Mirkin. Approximation problems in the space of relations and analysis of nonnumerical features. *Avtomat. i Telemekh.*, (9) :53–61, 1974. 110

[92] B. G. Mirkin. On the problem of reconciling partitions. In *Quantitative Sociology : Internations Perspectives on Mathematical and Statistical Modeling*, pages 441–449. Academic Press, New York, 1975. 110

[93] Guido Moerkotte and Thomas Neumann. Accelerating queries with group-by and join by groupjoin. *PVLDB*, 4(11) :843–851, 2011. 104, 150

[94] Richard A Mollin. *An introduction to cryptography*. CRC Press, 2006. 21

[95] Bernard Monjardet and Nathalie Caspard. On a dependence relation in finite lattices. *Discrete Mathematics*, 165 :497–505, 1997. 69, 119

[96] Bernard Monjardet and Vololonirina Raderanirina. Lattices of choice functions and consensus problems. *Social Choice and Welfare*, 23(3) :349–382, 2004. 111

[97] Peter Muth, Dirk Wodtke, Jeanine Weissenfels, Angelika Kotz Dittrich, and Gerhard Weikum. From centralized workflow specification to distributed workflow execution. *Journal of Intelligent Information Systems*, 10(2) :159–184, 1998. 12

[98] JB Nation. Finite sublattices of a free lattice. *Transactions of the American Mathematical Society*, pages 311–337, 1982. 113

[99] Sébastien Nedjar, Fabien Pesci, Lotfi Lakhal, and Rosine Cicchetti. The agree concept lattice for multidimensional database analysis. In *Formal Concept Analysis*, pages 219–234. Springer, 2011. 80

[100] Elsa Negre, Franck Ravat, Olivier Teste, and Ronan Tournier. Cold-start recommender system problem within a multidimensional data warehouse. In Roel Wieringa, Selmin Nurcan, Colette Rolland, and Jean-Louis Cavarero, editors, *RCIS*, pages 1–8. IEEE, 2013. 79

[101] Patrick E. O'Neil, Elizabeth J. O'Neil, Xuedong Chen, and Stephen Revilak. The star schema benchmark and augmented fact table indexing. In Raghunath Othayoth Nambiar and Meikel Poess, editors, *TPCTC*, volume 5895 of *Lecture Notes in Computer Science*, pages 237–252. Springer, 2009. 98

[102] Carlos Ordonez. Optimization of linear recursive queries in sql. *Knowledge and Data Engineering, IEEE Transactions on*, 22(2) :264–277, 2010. 162

[103] Oystein Ore. Theory of equivalence relations. *Duke Math. J.*, 9 :573–627, 1942. 11

[104] Oystein Ore. Theory of equivalence relations. *Duke Math. J.*, 9(3) :573–627, 1942. 118

[105] Lance Parsons, Ehtesham Haque, and Huan Liu. Subspace clustering for high dimensional data : a review. *ACM SIGKDD Explorations Newsletter*, 6(1) :90–105, 2004. 12

[106] David M Pennock, Eric Horvitz, C Lee Giles, et al. Social choice theory and recommender systems : Analysis of the axiomatic foundations of collaborative filtering. In *AAAI/IAAI*, pages 729–734, 2000. 110

[107] Jim Pitman. Exchangeable and partially exchangeable random partitions. *Probability theory and related fields*, 102(2) :145–158, 1995. 166

[108] R.C. Powers. Medians and majorities in semimodular posets. *Discrete Applied Mathematics*, 127(2) :325 – 336, 2003. Ordinal and Symbolic Data Analysis (OSDA '98), Univ. of Massachusetts, Amherst, Sept. 28-30, 1998. 111

[109] HA Priestley. Ordered sets and duality for distributive lattices. *North-Holland Mathematics Studies*, 99 :39–60, 1984. 69

[110] Hilary A Priestley. Ordered topological spaces and the representation of distributive lattices. *Proceedings of the London Mathematical Society*, 3(3) :507–530, 1972. 115

[111] Ken K. Pu and Alberto O. Mendelzon. Concise descriptions of subsets of structured sets. In *Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'2003)*, pages 123–133, 2003. 149

[112] Pavel Pudlák et al. Every finite lattice can be embedded in a finite partition lattice. *Algebra Universalis*, 10(1) :74–95, 1980. 11

[113] Yehuda Rav. Semiprime ideals in general lattices. *Journal of Pure and Applied Algebra*, 56(2) :105 – 118, 1989. 113

[114] Simon Régnier. Sur quelques aspects mathématiques des problèmes de classification automatique. *Mathématiques et Sciences humaines*, 82 :13–29, 1983. 110

[115] Klaus Reuter. The jump number and the lattice of maximal antichains. *Discrete Mathematics*, 88(2) :289–307, 1991. 133

[116] Theo JA Roelandt and Pim Den Hertog. Cluster analysis and cluster-based policy making in oecd countries : an introduction to the theme. *Boosting innovation : The cluster approach*, pages 9–23, 1999. 9

[117] Oscar Romero, Patrick Marcel, Alberto Abella, Veronika Peralta, and Ladjel Bellatreche. Describing analytical sessions using a multidimensional algebra. In Alfredo Cuzzocrea and Umeshwar Dayal, editors, *Data Warehousing and Knowledge Discovery*, volume 6862 of *Lecture Notes in Computer Science*, pages 224–239. Springer Berlin Heidelberg, 2011. 79

[118] Bertrand Russell. On the relations of universals and particulars. *Proceedings of the Aristotelian Society*, 12 :pp. 1–24, 1911. 9

[119] Bertrand Russell and Alfred North Whitehead. *Principles of Mathematics*. Cambridge University Press, 1910–1913. 10

[120] Domenico Sacca and Gio Wiederhold. Database partitioning in a cluster of processors. *ACM Transactions on Database Systems (TODS)*, 10(1) :29–56, 1985. 80

[121] Carsten Sapia. On modeling and predicting query behavior in olap systems. In *PROC. INT'L WORKSHOP ON DESIGN AND MANAGEMENT OF DATA WAREHOUSES (DMDW 99), SWISS LIFE*, pages 1–10, 1999. 79

[122] John E Shore and Rodney W Johnson. Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy. *Information Theory, IEEE Transactions on*, 26(1) :26–37, 1980. 10

[123] Yannis Sismanis and Nick Roussopoulos. The dwarf data cube eliminates the highy dimensionality curse. 2003. 79

[124] Venkataraman Srinivasan and Allan D Shocker. Linear programming techniques for multidimensional analysis of preferences. *Psychometrika*, 38(3) :337–369, 1973. 9

[125] Marshall H Stone. The theory of representation for boolean algebras. *Transactions of the American Mathematical Society*, 40(1) :37–111, 1936. 71, 115

[126] Alexander Strehl and Joydeep Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *The Journal of Machine Learning Research*, 3 :583–617, 2003. 109

[127] Shu-Hao Sun. On separation lemmas. *Journal of pure and applied algebra*, 78(3) :301–310, 1992. 71

[128] Robert E Tarjan and Jan Van Leeuwen. Worst-case analysis of set union algorithms. *Journal of the ACM (JACM)*, 31(2) :245–281, 1984. 183

[129] Robert Endre Tarjan. Efficiency of a good but not linear set union algorithm. *Journal of the ACM (JACM)*, 22(2) :215–225, 1975. 175

[130] Alfred Tarski. The concept of truth in formalized languages. *Logic, semantics, metamathematics*, 2 :152–278, 1956. 11

[131] Alexander Topchy, Anil K Jain, and William Punch. Clustering ensembles : Models of consensus and weak partitions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(12) :1866–1881, 2005. 109

[132] Jos Uffink. Can the maximum entropy principle be explained as a consistency requirement ? *Studies in History and Philosophy of Science Part B : Studies in History and Philosophy of Modern Physics*, 26(3) :223–261, 1995. 10

[133] Alasdair Urquhart. A topological representation theory for lattices. *Algebra Universalis*, 8(1) :45–58, 1978. 115, 137

[134] G. von Bultzingsloewen. Optimizing SQL queries for parallel execution. *ACM SIGMOD Record*, 18(4) :17–22, December 1989. 150

[135] Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schrödl, et al. Constrained k-means clustering with background knowledge. In *ICML*, volume 1, pages 577–584, 2001. 109

[136] Rudolf Wille. Subdirect decomposition of concept lattices. *Algebra Universalis*, 17(1) :275–287, 1983. 119

[137] Rui Xu, Donald Wunsch, et al. Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16(3) :645–678, 2005. 11

[138] Reza Bosagh Zadeh and Shai Ben-David. A uniqueness theorem for clustering. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 639–646. AUAI Press, 2009. 110

l'université nantes angers le mans
COMMUNAUTÉ D'UNIVERSITÉS ET ÉTABLISSEMENTS

UNIVERSITÉ DE NANTES

Thèse de Doctorat

Frédéric DUMONCEAUX

Approches algébriques pour la gestion et l'exploitation de partitions sur des jeux de données

Algebraic approaches for management and handling of set partitions over datasets

Résumé
L'essor des méthodes d'analyse de données dans des contextes toujours plus variés nécessite la conception de nouveaux outils permettant la gestion et la manipulation des données extraites. La construction de résumés est alors couramment structurée sous la forme de partitions d'ensembles dont la manipulation dépend à la fois du contexte applicatif et de leurs propriétés algébriques. Dans un premier temps, nous proposons de modéliser la gestion des résultats de requêtes d'agrégation dans un cube OLAP à l'aide d'un calcul algébrique sur des partitions. Nous mettons en évidence l'intérêt d'une telle démarche par le gain de temps et d'espace observé pour produire ces résultats. Nous traitons par la suite le cas de la modélisation du consensus de partitions où nous soulignons les difficultés propres à sa construction en l'absence de propriétés qui régissent la combinaison des partitions. Nous proposons donc d'approfondir l'étude des propriétés algébriques de la structure du treillis des partitions, en vue d'en améliorer la compréhension et par conséquent de produire de nouvelles procédures pour l'élaboration du consensus. En guise de conclusion, nous proposons la modélisation et une mise en œuvre concrète d'opérateurs sur des partitions génériques et nous livrons diverses expériences, propres à souligner l'intérêt de leur usage conceptuel et opérationnel.

Mots clés
algèbre, treillis, partition, relation de congruence, cube de données, consensus.

Abstract
The rise of data analysis methods in many growing contexts requires the design of new tools, enabling management and handling of extracted data. Summarization process is then often formalized through the use of set partitions whose handling depends on applicative context and inherent properties. Firstly, we suggest to model the management of aggregation query results over a data cube within the algebraic framework of the partition lattice. We highlight the value of such an approach with a view to minimize both required space and time to generate those results. We then deal with the consensus of partitions issue in which we emphasize challenges related to the lack of properties that rule partitions combination. The idea put forward is to deepen algebraic properties of the partition lattice for the purpose of strengthening its understanding and generating new consensus functions. As a conclusion, we propose the modelling and implementation of operators defined over generic partitions and we carry out some experiences allowing to assert the benefit of their conceptual and operational use.

Key Words
algebra, lattice, partition, congruence relation, data cube, consensus.