An order theoretic point-of-view on subgroup discovery

Aimene Belfodil

NNT : 2019LYSEI078

THÈSE DE DOCTORAT DE L'UNIVERSITÉ DE LYON
opérée au sein de
L'INSA DE LYON

ECOLE DOCTORALE N°512
MATHÉMATIQUES ET INFORMATIQUE (INFOMATHS)

Spécialité/Discipline de doctorat: Informatique

An Order Theoretic Point-of-view on Subgroup Discovery

À soutenir publiquement par
AIMENE BELFODIL
le 30/09/2019

Devant le jury composé de:

Pr. Bruno Crémilleux	Université de Caen	Rapporteur
Pr. Bernhard Ganter	Technische Universitaet Dresden	Rapporteur
Pr. Céline Robardet	INSA Lyon	Directrice de thèse
Dr. Mehdi Kaytoue	Infologic	Co-directeur de thèse
Dr. Peggy Cellier	INSA Rennes	Examinatrice
Pr. Miguel Couceiro	Université de Lorraine	Examinateur
Pr. Arno Siebes	Universiteit Utrecht	Examinateur
Pr. Sergei O. Kuznetsov	Higher School of Economics (Moscow)	Invité
Mr. Julien Zarka	Mobile Devices Ingenierie	Invité

Département FEDORA – INSA Lyon - Ecoles Doctorales – Quinquennal 2016-2020

SIGLE	ECOLE DOCTORALE	NOM ET COORDONNEES DU RESPONSABLE
CHIMIE	**CHIMIE DE LYON** **http://www.edchimie-lyon.fr** Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage **secretariat@edchimie-lyon.fr** INSA : R. GOURDON	M. Stéphane DANIELE Institut de recherches sur la catalyse et l'environnement de Lyon IRCELYON-UMR 5256 Équipe CDFA 2 Avenue Albert EINSTEIN 69 626 Villeurbanne CEDEX **directeur@edchimie-lyon.fr**
E.E.A.	**ÉLECTRONIQUE, ÉLECTROTECHNIQUE, AUTOMATIQUE** **http://edeea.ec-lyon.fr** Sec. : M.C. HAVGOUDOUKIAN **ecole-doctorale.eea@ec-lyon.fr**	M. Gérard SCORLETTI École Centrale de Lyon 36 Avenue Guy DE COLLONGUE 69 134 Écully Tél : 04.72.18.60.97 Fax 04.78.43.37.17 **gerard.scorletti@ec-lyon.fr**
E2M2	**ÉVOLUTION, ÉCOSYSTÈME, MICROBIOLOGIE, MODÉLISATION** **http://e2m2.universite-lyon.fr** Sec. : Sylvie ROBERJOT Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 INSA : H. CHARLES **secretariat.e2m2@univ-lyon1.fr**	M. Philippe NORMAND UMR 5557 Lab. d'Ecologie Microbienne Université Claude Bernard Lyon 1 Bâtiment Mendel 43, boulevard du 11 Novembre 1918 69 622 Villeurbanne CEDEX **philippe.normand@univ-lyon1.fr**
EDISS	**INTERDISCIPLINAIRE SCIENCES-SANTÉ** **http://www.ediss-lyon.fr** Sec. : Sylvie ROBERJOT Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 INSA : M. LAGARDE **secretariat.ediss@univ-lyon1.fr**	Mme Emmanuelle CANET-SOULAS INSERM U1060, CarMeN lab, Univ. Lyon 1 Bâtiment IMBL 11 Avenue Jean CAPELLE INSA de Lyon 69 621 Villeurbanne Tél : 04.72.68.49.09 Fax : 04.72.68.49.16 **emmanuelle.canet@univ-lyon1.fr**
INFOMATHS	**INFORMATIQUE ET MATHÉMATIQUES** **http://edinfomaths.universite-lyon.fr** Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage Tél : 04.72.43.80.46 **infomaths@univ-lyon1.fr**	M. Luca ZAMBONI Bât. Braconnier 43 Boulevard du 11 novembre 1918 69 622 Villeurbanne CEDEX Tél : 04.26.23.45.52 **zamboni@maths.univ-lyon1.fr**
Matériaux	**MATÉRIAUX DE LYON** **http://ed34.universite-lyon.fr** Sec. : Stéphanie CAUVIN Tél : 04.72.43.71.70 Bât. Direction **ed.materiaux@insa-lyon.fr**	M. Jean-Yves BUFFIÈRE INSA de Lyon MATEIS - Bât. Saint-Exupéry 7 Avenue Jean CAPELLE 69 621 Villeurbanne CEDEX Tél : 04.72.43.71.70 Fax : 04.72.43.85.28 **jean-yves.buffiere@insa-lyon.fr**
MEGA	**MÉCANIQUE, ÉNERGÉTIQUE, GÉNIE CIVIL, ACOUSTIQUE** **http://edmega.universite-lyon.fr** Sec. : Stéphanie CAUVIN Tél : 04.72.43.71.70 Bât. Direction **mega@insa-lyon.fr**	M. Jocelyn BONJOUR INSA de Lyon Laboratoire CETHIL Bâtiment Sadi-Carnot 9, rue de la Physique 69 621 Villeurbanne CEDEX **jocelyn.bonjour@insa-lyon.fr**
ScSo	**ScSo*** **http://ed483.univ-lyon2.fr** Sec. : Véronique GUICHARD INSA : J.Y. TOUSSAINT Tél : 04.78.69.72.76 **veronique.cervantes@univ-lyon2.fr**	M. Christian MONTES Université Lyon 2 86 Rue Pasteur 69 365 Lyon CEDEX 07 **christian.montes@univ-lyon2.fr**

*ScSo : Histoire, Géographie, Aménagement, Urbanisme, Archéologie, Science politique, Sociologie, Anthropologie

Publications presented in the Dissertation

The contributions presented in this thesis appear in the following publications:

International Conferences

- Aimene Belfodil, Sergei O. Kuznetsov, Céline Robardet and Mehdi Kaytoue. **Mining Convex Polygon Patterns with Formal Concept Analysis**. In the International Joint Conference on Artificial Intelligence, IJCAI, pages 1425–1432, 2017.
- Aimene Belfodil and Adnene Belfodil, Mehdi Kaytoue. **Anytime Subgroup Discovery in Numerical Domains with Guarantees**. In the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD), pages 500–516, 2018[1].
- Aimene Belfodil, Sergei O. Kuznetsov and Mehdi Kaytoue. **On Pattern Setups and Their Completions**. In Concept Lattices and Their Applications (CLA), pages 243–253, 2018.
- Aimene Belfodil, Adnene Belfodil and Mehdi Kaytoue. **Mining Formal Concepts using Implications between Items**. In the International Conference on Formal Concept Analysis (FCA), pages 173–190, 2019 (see `https://hal.archives-ouvertes.fr/hal-02080577/document`).

Submitted Papers in International Journals

- Aimene Belfodil, Sergei O. Kuznetsov and Mehdi Kaytoue. **On Pattern Setups and Pattern Multistructures**. Submitted to the International Journal of General Systems (IJGS) (See `https://arxiv.org/abs/1906.02963`).

Others

- Aimene Belfodil, Mehdi Kaytoue, Céline Robardet, Marc Plantevit and Julien Zarka. **Une méthode de découverte de motifs contextualisés dans les traces de mobilité d'une personne**. Extraction et Gestion des Connaissances (EGC), pages 63–68, 2016.
- Aimene Belfodil, Mehdi Kaytoue, Sergei O. Kuznetsov and Amedeo Napoli. **On Ordered Sets in Pattern Mining**. Tutorial at ECML/PKDD 2019 (See `https://belfodilaimene.github.io/pattern-setups-tutorial/`).
- Aimene Belfodil and Adnene Belfodil, Anes Bendimerad, Philippe Lamarre, Celine Robardet, Mehdi Kaytoue, Marc Plantevit. **FSSD - A Fast and Efficient Algorithm for Subgroup Set Discovery**. In the International Conference on Data Science and Advanced Analytics (DSAA), 2019.

[1] Rewarded by the ECML/PKDD'2018 award committee as the best student data mining paper of the year.

"... I can only say: je le vois, mais je ne le crois pas. ..."[2]
A Letter from Cantor to Dedekind, 29 June 1877 (**[62] pp. 860**)

It is with these terms that *Georg Cantor* announced his proof to *Richard Dedekind* of one of the fundamental results in Set Theory. I wanted to start this dissertation with this *Cantor* astonishment of his findings where I can imagine the *euphoria* he had when he started to understand such things after years of effort. It is such a kind of *euphoria* I am looking for ...

[2]"... I can only say: I see it, but I don't believe it. ..."

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

This monograph is entitled *"An Order Theoretic Point-of-view on Subgroup Discovery"* as it primarily concerns understanding and investigating notions of SUBGROUP DISCOVERY using the well-founded mathematical tools of ORDER THEORY.

SUBGROUP DISCOVERY (SD) falls into the quite large framework of *Knowledge Discovery in Databases* (*KDD*), a phrase coined by *Piatetsky-Shapiro* at the first *KDD* workshop in 1989 [142]. According to *Frawley* and *Piatetsky-Shapiro* [71], *"Knowledge discovery (in Data) is the nontrivial extraction of implicit, previously unknown, and potentially useful information from data"*. *KDD* is a process that, as the term may suggest, starts from the *data* and is supposed to converge at the end to some extracted *knowledge*. This process comprises the steps of data selection, data preprocessing, data transformation, *data mining* and *extracted "information" evaluation* [64] (see Fig. 1.1). In the last decades, many techniques have been proposed to automate this knowledge extraction from databases. *Subgroup Discovery*, or equivalently *Data Surveying*, represents one of these well-established techniques. Following Siebes [150], Subgroup discovery is the automatic task of discovering interesting subgroups in databases. By *subgroup*, it is meant a subset of rows in the database that is separable by a *description* using the attributes of the database, i.e. a query on the database using its schema. By *interesting* it is meant that one needs a formal way to assess the quality of the selected subgroup as for instance defining some quality measure that assigns to each subgroup a real value: the higher is this value the more interesting the corresponding subgroup is supposed to be. Before diving into more details, consider the following toy example.

Example 1.1. Let be some representative patient dataset where each patient (row) in the dataset is described by various properties: *age, gender, average number of cigarettes smoked per day, average duration of moderate-intensity physical activity per day, etc*. Along with these information, a label is added to each patient indicating if a lung cancer has been diagnosed for the patient or

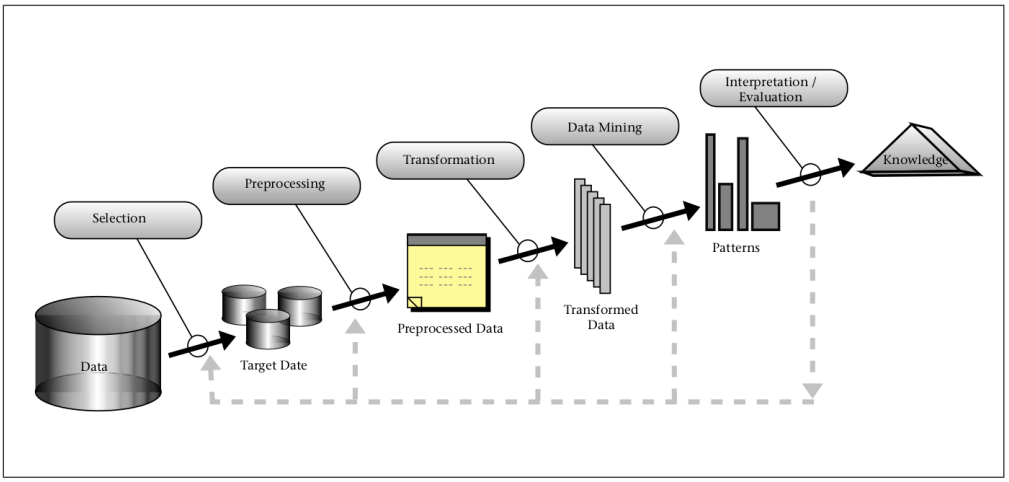


Figure 1.1: Steps of a KDD process (from [64])

not. Imagine that the prevalence of the lung cancer in this representative population of patients is 1 ‰ (i.e. 1 per thousands). An analyst who wants to *understand* what are the factors of risks of lung cancer from such a dataset can test many hypotheses using the properties of the patients and may come by the end with the following hypothesis:

"While the prevalence of lung cancer in the studied population is 1 °/₀₀*, the prevalence of lung cancer for men, aged between* 34 *and* 69 *years, smoking more than* 8 *cigarettes per day during the least* 10 *years and doing less than* 20 *minute per day of a moderate or vigorous intensity physical activity is equal to* 8 °/₀₀*"*

This latter hypothesis describes a *subgroup* where we have two parallel aspects: the *intent* aspect of subgroups representing the *description* using the different attributes of the dataset and the *extent* aspect representing no more than the patients falling under this description. Along, what made us think that the subgroup was *interesting* is the fact that the prevalence was 8 times higher than the normal prevalence when considering such a subgroup. One may suppose then that this hypothesis is a risk factor of the lung cancer. □[1]

Rather than testing the different hypotheses manually which is clearly an impossible task since the hypothesis space could be extremely huge, the analyst could use the automated technique of SUBGROUP DISCOVERY to find such subgroups as long as he[2] provides the two following parameters [150]:

1. The **Description Language**, or equivalently the **pattern language** or the **hypothesis space**[3], representing the space of hypotheses the analyst wants to explore in his study.

[1] The "□" signals the end of an example or a note.
[2] The "he" used in the manuscript is *ungendered*. It does hence not refer necessarily to a male person.
[3] Description Language, pattern language and hypothesis space will be used indistinguishably here.

2. The **Quality Measure** representing the way the analyst assesses the interestingness of the extracted hypotheses, e.g. the ratio between the prevalence in the dataset and the prevalence in the subgroup.

1.1 On Description Languages

As said earlier, the **description language** represents the search space of all possible hypotheses the analyst wants to explore to find interesting subgroups. We mean by a **subgroup** no more than a portion of the dataset, i.e. a subset of rows, that can be selected, **separated**, by a description in the considered description language. We have seen also in Example 1.1 a somehow realistic application of subgroup discovery. In order to introduce other notions more simply, let us study the following example.

Example 1.2. Consider the dataset depicted in Fig. 1.2 **(a)** of 10 elements $\{g_i\}_{1\leq i\leq 10}$. Each row is described by several properties: two numerical values x and y, a geometrical form in $\{\bigcirc, \Diamond, \square, \triangle\}$ and a color in $\{white, gray, black\}$. □

Several description languages can be used here to select subgroups in the dataset, i.e. subsets of elements from $\{g_i\}_{1\leq i\leq 10}$. We give below some examples:

(1) We consider descriptions that select subgroups by shape, i.e. $\{\bigcirc, \Diamond, \square, \triangle\}$. In such case, possible subgroups are $\{g_1, g_2, g_3\}$, $\{g_4, g_5\}$, $\{g_6, g_7, g_8\}$ and $\{g_9, g_{10}\}$ which are respectively induced by the following hypotheses: *" is a ○"*,*" is a ◊"*,*" is a □"* and *" is a △"*. Other subsets of objects like for instance $\{g_1\}$ or $\{g_1, g_2, g_4\}$ are not subgroups since they cannot be selected by the description language.

(2) If we add descriptions that combine properties of shapes rather than only the shapes themselves, other subsets of objects become separables. For instance, one can think about the two additional descriptions *"is a four-sided polygon"* and *"is a polygon"*, the subsets of objects $\{g_4, g_5, g_9, g_{10}\}$ and $\{g_4, ..., g_{10}\}$ become respectively separable by the language.

(3) If we combine properties of elements in the dataset, a lot more subsets of elements become again separable by the hypothesis space. For instance, if the property color, descriptions like the following one is added *"is a black circle"* which selects $\{g_1, g_2\}$. Such a language of combining properties from a set of properties by conjunction is commonly called **itemset pattern language** [4].

(4) Until now, we have ignored the numerical values x and y of each element of the database. Again, there are several ways to consider descriptions using properties x and y. As a matter of example, one can consider:

- Interval restrictions on x and y. For example, description *"*$2 \leq x \leq 7$ *and* $2 \leq y \leq 7$*"* depicted in Fig. 1.2 **(b)** selects subgroup $\{g_2, g_3, g_4, g_5, g_6, g_8, g_9\}$. Language of all of these restrictions is commonly called **interval pattern language** [104].

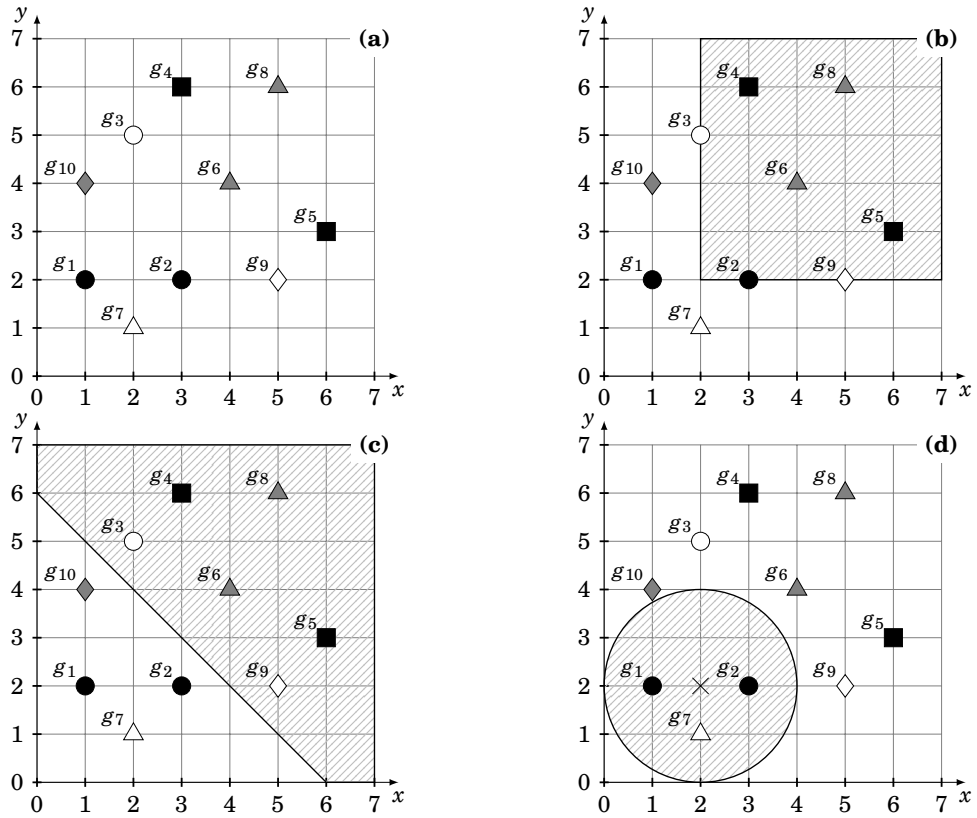


Figure 1.2: **(a)** A dataset of 10 elements described by the following properties: shape, color, x and y values. **(b)** A subgroup $\{g_2, g_3, g_4, g_5, g_6, g_8, g_9\}$ selected by *"*$2 \leq x \leq 7$ *and* $2 \leq y \leq 7$*"*. **(c)** A subgroup $\{g_3, g_4, g_5, g_6, g_8, g_9\}$ selected by *"*$x + y \geq 6$*"*. **(d)** A subgroup $\{g_1, g_2, g_7\}$ selected by *"*$(x-2)^2 + (y-2)^2 \leq 4$*"*.

- Linear inequalities using x and y. For instance, description *"*$x + y \geq 6$*"* depicted in Fig. 1.2 **(c)** selects subgroup $\{g_3, g_4, g_5, g_6, g_8, g_9\}$. One can see that such a subgroup cannot be selected using only interval restrictions on both x and y. Generally, subgroups separable by such a language are said to be *linearly separable*.
- Neighborhoods on the plane. For example, description *"On the vicinity of* $(2,2)$ *by a radius of* 2*"* or more formally *"*$(x-2)^2 + (y-2)^2 \leq 4$*"* depicted in Fig. 1.2 **(d)** selects subgroup $\{g_1, g_2, g_7\}$. The language regrouping all these disks is called **neighborhood pattern language** [86].

One should however keep in mind that the description language should stay interpretable and meaningful. This was well-explained by Boley[4]: *"The ultimate purpose of the first (Machine Learning) is automatization. i.e., to use a machine to substitute human cognition (e.g., for autonomously driving a car based on sensor inputs). In contrast, the purpose of the second (Subgroup Discovery) is to use machine discoveries to synergistically boost human expertise."* Hence a description

[4]Boley **realKD** Blog (`http://www.realkd.org/subgroup-discovery/`) - Last access 05 June 2019.

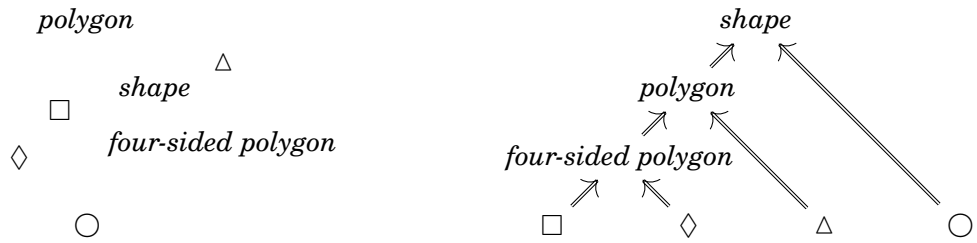


Figure 1.3: From an unordered description space **(left)** to an ordered description space using implications **(right)**.

language allowing the following form of description:

$$log(2 \cdot x + y^3) - \sqrt{2 \cdot x \cdot y} + y \leq 7621.1/e^{2 \cdot x^{2.3} - y}$$

should not be considered *a priori* as it is not comprehensible and thus violates the whole essence of subgroup discovery.

The state-of-the-art abounds with descriptions languages of different kinds depending on the analyzed dataset and the aim of the study. Indeed, besides the pattern languages that we have presented earlier, i.e. itemset [4], interval [104] and neighborhood [86] patterns; other languages were presented and investigated in the literature. We enumerate as a matter of example: (complex) sequential patterns [5, 39, 45, 143], graph patterns [110, 164], trajectory patterns [87], periodic patterns [74, 138], subgraph patterns in attributed graphs [23, 105].

1.2 Ordering Description Languages

Description languages can be seen just as a mere set of descriptions. However, such a point-of-view is clearly not helpful as it does not provide any information about the relationships descriptions may have between each other. Consider again Example 1.2 and the following questions:

(Q1) What is the relationship of being a *"four-sided polygon"* and being a *"square (□)"*?
(Q2) What is the relationship of being a *"square (□)"* and being a *"rhombus (◊)"*?

For question **(Q1)**, it is easy to see that all *"squares"* are *"four-sided polygons"*. In other words, there is a logical implication between these two properties. From the ORDER THEORY perspective, implications form an *order* between elements of the description language[5]. Fig. 1.3 **(right)** shows for instance how the description space $\{\bigcirc, \Diamond, \square, \triangle, \text{four-sided polygon}, \text{polygon}, \text{shape}\}$ is ordered thanks to implications. It does represent a hierarchy of concepts. Going back to Fig. 1.2, it is easy to see that these implications are simply represented by the inclusion relationship between the depicted geometrical forms. For example, if a rectangle R_1 is enclosed in rectangle R_2 (i.e. $R_1 \subseteq R_2$) then any point p falling into R_1 falls also in R_2 by definition (i.e. $p \in R_1$ implies $p \in R_2$).

[5]Technically speaking, this is only a pre-order. However, if logically equivalent descriptions are considered as the same hypothesis, this pre-order becomes a partial order. Formal Definitions are found in Chapter 2.

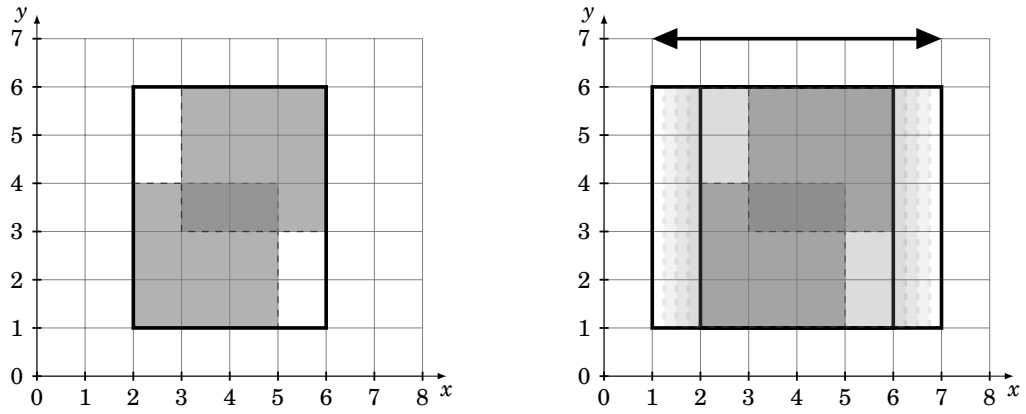


Figure 1.4: **(left)** The smallest enclosing rectangle $[2,6]\times[1,6]$ enclosing both $[2,5]\times[1,4]$ and $[3,6]\times[3,6]$. **(right)** The two squares $[2,5]\times[1,4]$ and $[3,6]\times[3,6]$ have not a unique smallest square enclosing them but an infinite set of minimal ones given by: $[1+l,6+l]\times[1,6]$ for any real number $l\in[0,1]$.

For question **(Q2)**, one can see that *"squares"* and *"rhombuses"* have in common the fact that they are *"four-sided polygons"*. In this second question, we have *composed* two descriptions in the hypothesis space to build a new description. Moreover, it is obvious that being a *"square"* or a *"rhombus"* implies being a *"four-sided polygon"*. This second answer shows that by identifying commonalities between descriptions, one can build new descriptions in the hypothesis space. If we use the ORDER THEORY terminology, this can be formalized as follow: the hypothesis *"four-sided polygons"* can be seen as the *join* (i.e. *least common subsumer*) of *"square"* and *"rhombus"* shapes in the ordered description space $\{\bigcirc,\Diamond,\square,\triangle,\text{four-sided polygon},\text{polygon},\text{shape}\}$ by implications. Moreover, this latter ordered description space is said to be a *join-semilattice* as this *least common subsumer* exists and is unique for each pair of descriptions. This latter property holds in many description spaces but is not trivial as shown in the following example.

Example 1.3. Consider now the description language of rectangles structured by inclusion or equivalently the language of interval restrictions over attributes x and y structured by implications. Let be the two gray rectangles depicted in Fig. 1.4 **(left)**. The smallest (w.r.t. $\subseteq$) rectangle (the join, the least common subsumer) enclosing both $[2,5]\times[1,4]$ and $[3,6]\times[3,6]$ exists, is unique and is given by $[2,6]\times[1,6]$ depicted by the white enclosing rectangle in Fig. 1.4 **(left)**.

Now, let be the description language of all **squares** structured by inclusion. The two gray squares $[2,5]\times[1,4]$ and $[3,6]\times[3,6]$ depicted in Fig. 1.4 **(right)** have not a unique smallest square enclosing them but an infinite set of minimal ones given by $[1+l,6+l]\times[1,6]$ for any real number $l\in[0,1]$ (i.e. the moving square in Fig. 1.4 **(right)**). Hence, we say that this language does not form a join-semilattice. □

One should note that ORDER THEORY provides techniques to transform ordered description languages that are not join-semilattices to lattices by adding (missing) elements. Such transformations are usually called *completions* (e.g. Dedekind-MacNeille Completion [53]). For instance,

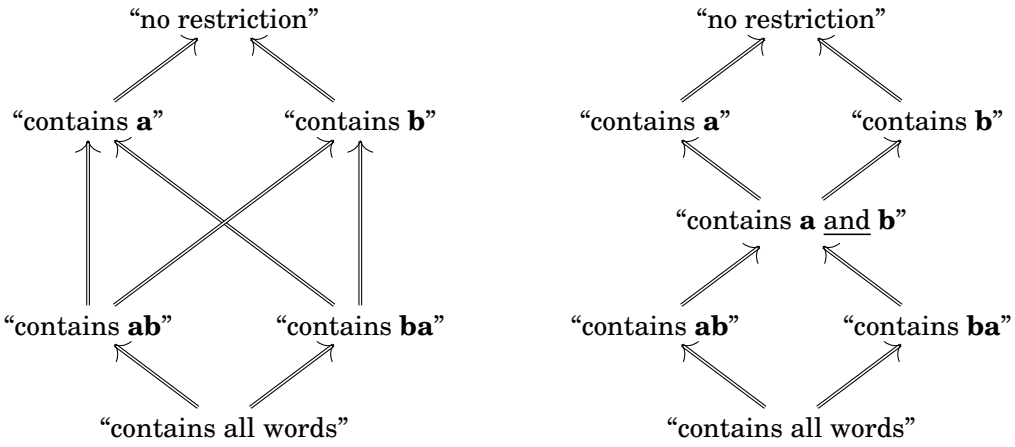


Figure 1.5: A non-lattice description language ordered by implications **(left)** and its *completion* to a lattice **(right)** thanks to adding the description *"contains **a** and **b**"*.

adding the intersections of squares in the description language of squares builds the description language of rectangles. Another example of the completion of a finite description language is provided below.

Example 1.4. Let be the description space ordered by implications depicted in Fig. 1.5 **(left)**. Imagine that these hypotheses are used in a dataset of words built over the alphabet $\{a, b\}$. Obviously, if a word contains *"ab"* then it does contain both *"a"* and *"b"*. Clearly, one can see that this ordered description language does not form a join-semilattice. Indeed, *"containing ab"* and *"containing ba"* concepts do not have a least common subsumer, i.e. a join. They have however two minimal common subsumers which are *"containing a"* and *"containing b"*.

Adding however the (missing) description *"contains a and b"* to the description space depicted in Fig. 1.5 **(left)** allows to build the lattice depicted in Fig. 1.5 **(right)**. □

1.3 From Datasets to Interesting Subgroups

We have discussed earlier how description languages participate to select subgroups in datasets intelligibly and how they can be structured. However, what is an interesting subgroup (or more generally a hypothesis) in a dataset?

Consider again the dataset depicted in Fig. 1.2 and the following question:

(Q3) What is the relationship of being a *"square (□)"* and being *"black"*?

Independently from the dataset, it seems that there is no relationship between being a *"square (□)"* and being *"black"*. However, one can see that particularly in this dataset, all rectangles (g_4 and g_5) are black. This latter implication arises from the studied dataset rather than the hypothesis space itself as it was the case for the two questions **(Q1)** and **(Q2)**. We say that

these implications are **informative** as they may provide relevant information on the potential knowledge hidden in the dataset.

Identifying informative implications in datasets has been thoroughly studied in the literature where it is somehow hard to trace back the beginning. For instance, analyzing implications appearing in Boolean datasets representing surveys can be found in [69] published in 1965 where ORDER THEORY and LATTICE THEORY already played an essential role. Later, in 1982, *Wille* [162] provided a new equivalent vision between Boolean datasets and complete lattices and refered to Boolean datasets as **formal contexts**. Such a vision, dubbed Formal Concept Analysis (FCA) [80], showed its potential as a tool for data analysis and data surveying. As a matter of example, Duquennes and Guigues presented in 1986 [91] the smallest set of informative implications existing in a Boolean dataset.

While this field of Formal Concept Analysis was flourishing, another field appeared from the database community in the early nineties and is commonly identified as the seminal paper of pattern mining (Agrawal et al. [4]). The aim of this work was to identify **association rules** in Boolean datasets representing customer transactions in supermarkets. Association rules $X \Rightarrow Y$ can be seen as relaxed informative implications in databases, that is if the *premise* X of the association rule occurs on a row in the dataset then there is a high chance that Y occurs also on the same row. For instance, if we consider the dataset depicted in Fig. 1.2 **(a)**, "$\bigcirc \Rightarrow$ *black*" is not an implication. However, if the description "$\bigcirc$" occurs in an element of the dataset, there is 66% of chance that the element is also "*black*". We say that the association rule "$\bigcirc \Rightarrow$ *black*" has a confidence of 66%. Implications are then just a *"100%"* confidence association rules. One should note that association rules where called **partial implications** by Luxenburger in 1991 [124]. While the connection between formal concept analysis and pattern mining seem today evident, the connection between these two fields was made independently by Zaki et. al [169] and Pasquier et al. [139] in the late nineties when they popularized the notions of **closed patterns** and **Galois connections**.

Beside exact and partial implications, the beginning of the era on pattern mining was often related to frequent patterns as they played a crucial role to look for association rules. Frequent patterns are simply pattern occurring in a non-negligible part of the dataset. Giacometti. et al. [85] identified this enthusiasm (or craze) about frequent patterns as the *"obsession of frequency"*. Later, other tasks attracted more attention. One of these tasks that we will investigate in more details in this thesis is what we call **discriminative subgroup discovery**. This latter task is defined as follow: *"given a target population, i.e. a subset of elements in a given dataset (e.g. patients having lung cancer in Example 1.2), identify subgroups where the target population is significantly over-represented comparing to the whole dataset"*. Different tasks falls under this field, we enumerate: JSM[6] methods [67, 68] , contrast pattern mining [13], emerging patterns [57], and the particular task of subgroup discovery developed in Section 2.4 in [163]. Class

[6]JSM methods were named after the English philosopher John Stuart Mill, who introduced methods of inductive reasoning in 19^{th} century according to Kuznetsov [111].

association rules (CAR) [121] and Inductive logic programming [133] are also tightly linked to discriminative subgroup discovery as they aim to provide good and comprehensible rules for classification. Please note that what we call here *discriminative subgroup discovery* was called *supervised descriptive rule discovery* by Novak et al. [134]. We do prefer here using *discriminative subgroup discovery* to emphasis the fact that this latter task falls under subgroup discovery and not the converse.

Other pattern mining tasks have emerged since then. We cite for example: Exceptional Model Mining (EMM) [117], Redescription Mining [75, 147] and High Utility Pattern Mining [70]. The most important commonality in these different aforementioned tasks is the fact that someone needs some **quality measure** to gauge the interestingness of the hypothesis following Siebes [150] and Wrobel [163].

1.4 Enumeration and Subgroup Discovery

We have explained earlier that the analyst behind the subgroup discovery task needs to define what is the **description language** he wants to explore and how he gauges the interestingness of hypotheses using some **quality measure**. Once these two aspects are defined, the automation of the subgroup discovery requires in its core **enumeration algorithms**.

1.4.1 Finding all Subgroups

Let us ignore for now the second aspect of subgroup discovery of how interestingness is gauged and focus on the task of enumerating all subgroups in a given dataset for a pre-defined hypothesis space. The easiest way is to exhaustively enumerate all elements of the hypothesis space and then to see what are the objects matching each description. Such an enumeration technique is potentially redundant and computationally expensive as it can output the same subgroup twice. For instance, reconsider the dataset depicted in Fig. 1.2 **(a)** and consider a description space where containing *"black rectangle"* and *"rectangle"*. These two descriptions induce the same subgroup $\{g_1, g_2\}$. Hence, visiting these two hypotheses generate the same subgroup twice. The only way to reduce the computational effort to look for all possible subgroups induced by some description language is to consider some subset of descriptions in the pattern language where it is guaranteed that all subgroups induced by the pattern language are also induced by this subset.

Closed patterns played an essential role to enumerate the set of all possible subgroups exhaustively with less computational effort. Indeed, when we consider the language of itemsets [4, 139], there is a one-to-one correspondence between all possible subgroups and the set of closed patterns. Hence, for this kind of patterns, the closed patterns represent the smallest subset of descriptions from which all subgroups can be induced. Therefore, enumerating exhaustively and non-redundantly all subgroups induced by the hypothesis space becomes equivalent to enumerating exhaustively and non-redundantly closed patterns. This later kind of enumeration

was thoroughly investigated in the literature since it is equivalent to enumerating fixpoints of some closure operator. One can cite Ganter's NEXTCLOSURE Algorithm [76], BORDAT's Algorithm [33] and CLOSE-BY-ONE (CBO) Algorithm [109, 110] among others. We invite the reader to see Kuznetsov and Obiedkov [113] surveying such techniques. Following the same spirit, subgroups induced by interval patterns can be enumerated exhaustively and non-redundantly using closed interval patterns [104] thanks to an adaptation of CLOSE-BY-ONE (CBO) Algorithm [109, 110].

Closed patterns have then been generalized to other types of pattern languages as for instance sequential patterns [166] and graph patterns [165]. Such closed patterns provide again a small subset of description from which all subgroups can be deduced. However, it should be noted that there is no longer a one-to-one correspondence between closed patterns and subgroups for these two latter languages. In fact, this problem is tightly linked to the fact that these two pattern languages are not lattices [168] as it is the case for itemset and interval patterns.

1.4.2 Finding Interesting Subgroups

Consider now the task of subgroup discovery where we aim to find a small subset of interesting subgroups w.r.t. to some **quality measure**, say the *top-k* subgroups [163]. The naivest way is to use the *brute-force algorithm*, that is: (1) explore and evaluate all hypotheses in the description language, (2) output at the end the top-k subgroups. However, such a technique can be intractable (if not impossible) when the hypothesis space grows. In order to do less effort to find the top-k subgroup, someone needs to skip visiting some (unpromising) hypotheses while having some information about them using the earlier visited descriptions. To do so, one needs to leverage the properties of the considered descriptions language on the one hand and the properties that the quality measures on the other hand. For instance, if the quality measure evaluation depends solely on the subgroup and not on its description, one can consider only closed patterns as they induce all possible subgroups (see Section 1.4.1). However, this may be not sufficient as someone needs still to visit all subgroups to find the *top-k* ones.

Other approaches flourished in the literature to ensure finding the interesting subgroups more efficiently. We cite, measures properties ((Anti-)Monotone, Convertible (Anti-)Monotone, Loose (Anti-)Montone, Piecewise (Anti-)Monotone, etc.) [32, 47] and optimistic estimates [89, 90, 132] as a matter of example.

These various techniques ensure finding the top-k possible subgroups and are said to be **exact**. However, they may also become intractable when the considered search space are of important size. In response to that, other types of algorithms emerged in the literature known to be **non-exact**. We cite for example Beam Search Techniques [59, 115, 159, 160], Evolutionary Algorithms [44], Sampling techniques [23, 30, 31] and Anytime algorithms [34].

1.5 Contributions

1.5.1 Patterns as Ordered Sets

Modern order and lattice theory provide convenient mathematical tools for pattern mining. Different formal tools have been proposed in the literature to model description languages. Among them, Formal Concept Analysis (FCA) [80] provides a natural and a well-founded mathematical tool to analyze binary datasets (i.e. contexts). However, since real-world datasets come in general with various complex attributes (e.g. numerical or nominal ones) rather than only Boolean ones, they need to be transformed to contexts (i.e. boolean datasets) before any manipulation using FCA. This kind of transformation has been proposed by [79] under the term of *(conceptual) scaling* (i.e. *binarizing*). Yet, even if *scaling* is a quite general tool, *binarizing* a dataset with regard to some pattern search spaces is not always obvious [11, 20]. In response to that, some other more natural tools to formalize complex description languages have been proposed. One could cite Logical Concept Analysis (LCA) proposed by Ferré and Ridoux [66], Pattern Structures proposed by Ganter and Kuznetsov [78] and Symbolic Data Analysis (SDA) [36, 37]. Pattern Structures allow for instance to model in a quite natural way many description languages. Indeed, itemset [4], interval [104], partition [12] pattern spaces among others [78, 112] can be modeled within the pattern structure framework.

Nevertheless, since pattern structures rely on meet-semilattices[7], some pattern spaces that are only partially ordered sets (posets) cannot be "directly" defined using such a framework. For instance, the *"sequences of itemsets"* pattern language [6] ordered by *"is subsequence of"* does not form a meet-semilattice [48, 168]. The sequential meet-semilattice [39, 48], or equivalently conjunctive sequence patterns [146] or the space of closed patrial orders [45] refer usually to *antichains (i.e. conjunction) of sequences* rather than to the poset of *sequences* itself. Same holds for the graph meet-semilattice from [78, 110]. In fact, these description languages are transformed to lattices before manipulating them. An example of such a transformation is shown in Example 1.4 and Fig. 1.5.

Surprisingly, studying pattern languages as ordered sets without any additional property and without transforming them to lattices have received a very little attention in the literature. In 2015, Lumpe and Schmidt [123] proposed such a framework under the name of **pattern setup**. However, they have not investigated it in more details as the aim of their paper was to study morphisms between pattern structures.

Contribution 1. In this first contribution, we endeavor to provide a better understanding of the pattern setup framework. We show that while pattern structures demand a strong condition on the posets of patterns (i.e. upper-bounded meet-semilattices), pattern setups do not require any

[7] We have presented earlier in Section 1.2 join-semilattices when the description languages were ordered by implications. This notion is in fact dual to meet-semilattices since description languages are commonly ordered by subsomptions which are the converse of implications.

additional property on the description space, which makes them rather permissive. Simply put, objects in a pattern setup could share some common descriptions in the pattern space but none of them is maximal (or equivalently closed) w.r.t. the subsomption order. One direct resulting problem of such an observation, is that closed descriptions do no longer induce all the subgroups separable by the description language.

To solve this issue, we require that the set of maximal common descriptions resume properly the set of common descriptions of any subset of objects. This is done by introducing the framework of pattern multistructure, a framework that demand the description language to be a meet-multisemilattice [22, 50, 128] rather than a meet-semilattice. This framework is far more permissive than pattern structures and allows to models the sequential and graph pattern languages. Moreover, closed descriptions in pattern multistructures induce all the subgroups separable by the pattern language.

We show also that the usual transformations used in the literature to transform pattern setups to pattern structures, as it is the case for sequential and graph pattern languages [39, 78, 146], are tightly linked to poset *completions*. For instance, enriching the description language with the set of maximal common descriptions is basically an Antichain completion (see [27, 53]). Such a transformation is applicable on a pattern setup, i.e. builds a pattern structure, if and only if the pattern setup is a pattern multistructure.

A preliminary version of this contribution has appeared in the proceedings of the Concept Lattices and their Applications conference (CLA'18) [18]. A longer and more detailed version has been submitted to the International Journal of General Systems (IJGS) and is currently under review [19].

1.5.2 Subgroup Exhaustive Enumeration

After modeling the pattern languages, we will focus in this work by the task of determining the set of all possible subgroups induced by a description language. Particularly, we investigate pattern languages which order induces a lattice and hence a pattern structure when finite datasets are considered. We consider two particular problems.

The first problem is related in subgroup enumeration in numerical datasets. Often, for this kind of datasets, the language of interval patterns is considered, i.e. the language of conjunction of interval restrictions over the numerical attributes (e.g., pattern $50 \leq age < 70 \wedge smoke\_per\_day \geq 3$). Kaytoue et al. [104] proposed an algorithm for enumerating exhaustively and non-redundantly all subgroups induced by the interval pattern language. However, exhaustive enumeration of subgroups induced when considering other numerical pattern languages have surprisingly not attracted much research interest.

Contribution 2. As a second contribution, we investigate the pattern language of conjunctions of linear inequalities over the numerical attributes of the dataset (e.g. conjunctions of patterns of the form $5 \cdot smoke\_per\_day + age \geq 3$). This language is interestingly equivalent to the language

of convex polytopes. We address the problem using pattern structures [78] and we present three exhaustive and non-redundant algorithms to enumerate the set of all possible subgroups induced by such a language. This contribution has appeared in the proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'17) [20].

The second problem is related in subgroup enumeration when Boolean datasets (contexts) are considered. We investigate particularly how the enumeration of subgroups or equivalently closed itemsets in a context can be enhanced when the inherent implications between items are considered. Inherent Implications between items can be seen in, but not only in, scaled contexts [79], i.e. contexts that are the result of binarization of some complex attribute as numerical ones. For instance, a numerical attribute whose values are $\{1,2,3\}$ can be transformed to a set of items $\{\leq 1, \leq 2, \leq 3, \geq 3, \geq 2, \geq 1\}$ thanks to *interordinal scaling* where we can see the inherent implications between items "≤ 1 implies ≤ 2", "≤ 2 implies ≤ 3" and so on.

Many algorithms have been proposed to enumerate exhaustively and non-redundantly the set of all possible subgroups [8, 33, 76, 109, 113, 136] by leveraging a closure operator. However, these standard algorithms do not take advantage of the potential relationships between attributes as for example "≤ 1 implies ≤ 2". For such, they perform additional closure computations which substantially degrade their performance. Particular instances of contexts with implications where handled in the literature. For instance, interordinal scaled contexts are directly linked to interval patterns as investigated by [104]. Analogously, ordinal scaled contexts are linked to datasets augmented with a taxonomy (i.e hierarchy) between items (e.g. *rectangles* are *four-sided polygons*) [17, 46]. Yet, when a context is provided with an arbitrary set of implications (i.e. forming some directed graph between items), no generic algorithm is provided.

Contribution 3. As a third contribution, we propose a generic algorithm, named CbOI for Close-by-One using Implications, to enumerate subgroups using the inherent implications between items provided as an input. In other words, provided a pair (context, directed graph of implications between attributes), CbOI uses at its best the provided implications between items to enumerate exhaustively and non-redundantly all subgroups. The proposed algorithm relies on a *Divide & Conquer* scheme to enumerate closed sets in a strongly accessible set system [29, 83]. We show that, in fact, closed itemsets are upper ideals (i.e. upward closed) in an equivalent poset of the input directed graph. We use then the fact that the set of these upper-ideals forms a strongly accessible set system. Building on these notions, we elaborate algorithm CbOI. This contribution has appeared in the proceedings of the International Conference on Formal Concept Analysis (ICFCA'2019) [16].

1.5.3 Anytime Discriminative Subgroup Discovery

The third part of this work considers the problem of discovering patterns that accurately discriminate one class label from the others in a labeled numerical dataset. This can be seen as

a particular task of subgroup discovery [106, 150, 163] dubbed here discriminative subgroup discovery.

As we have said earlier, when it comes to numerical attributes, a pattern is generally a conjunction of interval restrictions over the attributes, e.g., pattern $50 \leq age < 70 \wedge smoke\_per\_day \geq 3$ fosters lung cancer incidence. To look for such patterns (namely interval patterns), various approaches are usually implemented. Common techniques perform *a discretization* transforming the numerical attributes to categorical ones in a pre-processing phase before using the wide spectrum of existing mining techniques [10, 30, 122, 160]. This leads, however, to a loss of information even if an exhaustive enumeration is performed on the transformed data [10]. Other approaches explore the whole search space of all restrictions either exhaustively [40, 89, 104] or heuristically [34, 125]. While an exhaustive enumeration is generally unfeasible in large data, the various state-of-the-art algorithms that heuristically explore the search space provide no provable guarantee on how they *approximate* the top quality patterns and on how far they are from an exhaustive search. Recent techniques set up a third and elegant paradigm, that is direct sampling approaches [30, 31, 86]. Algorithms falling under this category are non-enumerative methods which directly sample solutions from the pattern space. They simulate a distribution which rewards high quality patterns with respect to some interestingness measure. While in [30, 31], authors propose a direct two-step sampling procedure dedicated for categorical/boolean datasets, authors in [86] devise an interesting framework which add a third step to handle the specificity of numerical data. The proposed algorithm addresses the discovery of dense neighborhood patterns by defining a new density metric. Nevertheless, it does not consider the discovery of discriminant numerical patterns in labeled numerical datasets. Direct sampling approaches abandon the completeness property and generate only approximate results. In contrast, anytime pattern mining algorithms [34, 98] are enumerative methods which exhibits the anytime feature [172], a solution is always available whose quality improves gradually over time and which converges to an exhaustive search if given enough time, hence ensuring completeness. However, there is no algorithm proposed so far for the particular task of discriminative subgroup discovery that ensure some guarantees on the outputted subgroups upon interruption.

Contribution 4. In this fourth contribution, we tackle the problem of providing an anytime algorithm that provide guarantees on the outputted subgroups upon interruption and ensures completeness if enough time is given. The proposed anytime algorithm, namely REFINEANDMINE, is tailored for discriminant interval patterns discovery in numerical datasets. It starts by mining subgroups in a coarse discretization, followed by successive refinements yielding increasingly finer discretizations highlighting potentially new interesting subgroups. Eventually, it performs an exhaustive search, if given enough time. Additionally, REFINEANDMINE gives two provable guarantees at each refinement. The first evaluates how close is the best found subgroup so far to the optimal one in the whole search space. The second measures how already found subgroups are diverse and cover well all the interesting regions in the dataset. This contribution has appeared

in the proceedings of The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD'2018) [15].

1.6 Structure of the Thesis

This work consists of three parts presented below.

The first part consists of two chapters (2 and 3). Chapter 2, entitled *order-theoretic foundations*, presents and discusses the various mathematical tools provided by ORDER THEORY and LATTICE THEORY to understand pattern languages. An interesting fact that we have discovered while writing this section is the connection that exists between complete multilattices [22, 50, 128, 129], antichain completions [27] and chain-complete posets. A reader who knows what is understood by a lattice and order in mathematics may skip this chapter. We advise him however to take a look on multilattices as it is a non commonly studied property. Next, Chapter 3 presents our first contribution. We start by presenting formal concept analysis [80, 162] as a first point of view on pattern languages. Then, we study the most generic tool that we will manipulate in this thesis which is pattern setups [123]. Depending on the properties of the ordered sets of descriptions, other structures stem and are discussed, i.e. pattern structures [78] and pattern multistructures [18]. The writing of this chapter relies on [18] and its extended version submitted to the International Journal of General System (IJGS'19) currently under review [19].

The second part consists of two chapters (4 and 5). Chapter 4 introduces the notion of set systems and enumeration algorithms in a general way and Chapter 5 investigates some techniques to enumerate exhaustively and non-redundantly subgroups induced by some pattern languages. We discuss here also our second [20] and third [16] contributions.

The third and the last part of the thesis consists also of two chapters (6 and 7) where we will be interested in the particular task of discriminative subgroup discovery. Chapter 6 presents this task in a general way while Chapter 7 presents our fourth contribution. The writing of this last chapter relies on [15].

Finally, Chapter 8 gives the conclusion and the future perspectives of this thesis.

Part I

Patterns as Ordered Sets

CHAPTER 2

ORDER-THEORETIC FOUNDATIONS

This chapter is entitled *"Order-Theortic Foundations"* following exactly the title of the first chapter in R. Wille and B. Ganter's book on Formal Concept Analysis [80]. The aim here is to provide the basic mathematical foundation to formalize and study pattern languages and their properties. The writing of this chapter was inspired partly from [80, 144, 148] and is organized as follow:

- **Section 2.1** recalls some basic facts on set theory.
- **Section 2.2** details the basic elements of order theory.
- **Section 2.3** presents some particular properties on partially ordered sets. We investigate in this section particularly the properties of lattices (subsection 2.3.4) and multilattices (subsection 2.3.5). Multilattices are important structures that we will use later in Chapter 3. We make also in this section a new connection between the property of chain-completeness and the property of being a complete multilattice.
- **Section 2.4** presents transformations of posets using morphisms. We will mainly be interested here in Galois connections, closure and kernel operators on complete lattices as they are important mappings for enumerating patterns. We take also the opportunity here to discuss completions (e.g. Dedekind-MacNeille completion). Particularly, we study the antichain embedding of posets and its relationship with the property of being a complete multilattice in subsection 2.4.4.3. Such an embedding is usually used in Formal Concept Analysis to transform non-lattice pattern languages to lattice-pattern languages that we will study further in details in Chapter 3.

2.1 Elements of Set Theory

2.1.1 Basic Definitions

A set X is an arbitrary collection of (distinct) elements. If an element x is in X, we denote $x \in X$, otherwise we denote $x \notin X$. For two sets S and X, S is said to be a **subset** of X and we denote $S \subseteq X$ iff $(\forall x \in S)\, x \in X$. Sets could be **finite** or **infinite**, the **size** (**cardinality**) of a finite set X is denoted $|X|$ and consists in the number of its elements. Particularly, the **empty set**, denoted $\emptyset$, is the unique set which size is 0. The **powerset** of X, is the set denoted $\wp(X)$ and given by the set of all subsets of X:

$$\wp(X) = \{S \mid S \subseteq X\}$$

Let X and Y be two nonempty sets, the **cartesian product** is the nonempty set:

$$X \times Y = \{(x, y) \mid x \in X \text{ and } y \in Y\}$$

2.1.2 Binary Relations

Sets can be linked by relations. A **binary relation** between two sets X and Y is any subset $R \subseteq X \times Y$. If the pair $(x, y) \in R$ we say that x is **in relation** (*in this direction*) with y and we denote $x\ R\ y$. Otherwise, we denote $x \not R\ y$ (i.e. $(x, y) \notin R$).

Functions. Important binary relations are **functions** or **mappings**. A **function** f from X to Y is a binary relation in $X \times Y$ that associates to each $x \in X$ a unique element $y \in Y$ which is denoted $f(x)$. We denote $f : X \to Y$. For any subset $S \subseteq X$, we call the **image** of S, the set denoted $f[S]$ and given by:

$$f[S] = \{f(x) \mid x \in S\}$$

A function f is said to be **surjective** if $f[X] = Y$, it is said to be **injective** if two distinct elements are always associated to two distinct values (i.e. $(\forall x_1, x_2 \in X)\ x_1 \neq x_2 \Rightarrow f(x_1) \neq f(x_2)$). In case where f is **injective** and **surjective**, we say that f is a **bijection** (a **one-to-one correspondance**) and that X and Y are **isomorphic**. The set of all possible functions from X to Y is denoted Y^X.

One should note that when we write for some set I, $\{x_i \mid i \in I\}$. The set I is called an **Index set** and there is technically a function from I to some set X that associates to each **indice** $i \in I$, an element x_i from X. Notice that for $i \neq j \in I$, one could have $x_i = x_j$.

Binary Relation on a Set. A binary relation R on a set X is any $R \subseteq X \times X$. It is said to be:

- **Reflexive** if $(\forall a \in E)\ a\, R\, a$.
- **Transitive** if $(\forall a, b, c \in E)$ if $a\, R\, b$ and $b\, R\, c$ then $a\, R\, c$.

- **Symmetric** if $(\forall a, b \in E)$ if $a\,R\,b$ then $b\,R\,a$.
- **Anti-symmetric** if $(\forall a, b \in E)$ if $a\,R\,b$ and $b\,R\,a$ then $a = b$.

Binary relations having two or more of these properties build special relations. We call:

- a **Preorder** any relation that is reflexive and transitive.
- a **Tolerance** any relation that is reflexive and symmetric.
- an **Equivalence** a relation that is reflexive, transitive and symmetric.

For an **equivalence** relation $\leftrightarrow$ on X, we draw the reader's attention to the two following notions. The **equivalence class** of $a \in X$, denoted $\overline{a}$, is the set of its equivalent element in X. It is given formally by $\overline{a} := \{b \in X \mid b \leftrightarrow a\}$. The **quotient set** of X by $\leftrightarrow$, denoted $X/\leftrightarrow$, is the partition[1] of X on equivalent classes; i.e. $X/\leftrightarrow := \{\overline{a} \mid a \in X\}$.

New Relations from the Old ones. One can also creates some relations from the others. Let R be a binary relation on a set X and let $S \subseteq X$. We call:

- The **restriction** of R onto S, the binary relation $R \cap S \times S$. Relation $R \cap S \times S$ is said to be a **sub-relation** of R. The restriction preserves the four aforementioned properties.
- The **dual relation** or the **inverse relation** of R, denoted R^{-1}, the following relation: $(\forall a, b \in X)\ a\,R^{-1}\,b \Leftrightarrow b\,R\,a$. The dual relation preserves the four aforementioned properties.
- The **reflexive closure** of R, denoted $R^{=}$, the smallest reflexive relation containing R. Formally: $R^{=} := R \cup \{(e,e) \mid e \in E\}$.
- The **transitive closure** of R, denoted R^{T}, the smallest transitive relation containing R.

2.1.3 On the Axiom of Choice (AC)

Until now, we have presented tools using only the basic axiomatization of set theory (i.e. **Zermelo-Fraenkel set theory (ZF)**). However, we will sometime require the usage of an additional axiom named the **axiom of choice (AC)** which help us in general to prove the existence of some elements without necessarily knowing how to build them.

Definition 2.1 (Axiom of Choice (AC)). Let $\mathcal{F} = \{S_i \mid i \in I\}$ be a nonempty family of nonempty sets. There exists a function:

$$f : \mathcal{F} \to \bigcup_{i \in I} S_i$$

such that $f(S_i) \in S_i$ for all $i \in I$. Such a function is called a **choice function**. Intuitively, it does choose a single element from each subset S_i.

The axiom of choice helps for instance to build the Cartesian product of any family of nonempty sets and show that it will be not empty. The **Cartesian product** of a nonempty family $\mathcal{F} = \{S_i \mid i \in I\}$ of nonempty sets is given by:

$$\bigtimes_{i \in I} S_i = \{(x_i)_{i \in I} \mid (\forall i \in I)\, x_i \in S_i\}$$

[1] A **partition** P on X is a set $P \in \wp(X)$ such that $\forall S_1, S_2 \in P$: $S_1 \cap S_2 = \emptyset$ and $\bigcup P = X$

2.2 Partially Ordered Sets

2.2.1 Basic Definitions

Definition 2.2. A **partial order** on a set P is a binary relation $\leq$ on P that is reflexive, transitive and anti-symmetric. The pair $(P, \leq)$ is called a **partially ordered set** or a **poset**. A poset is said to be finite if P is **finite**. Otherwise, it is said to be **infinite**. Two elements x and y from P are said to be **comparable** if $x \leq y$ or $y \leq x$; otherwise, they are said to be **incomparable**. A subset $S \subseteq P$ is said to be a **chain** (resp. an **antichain**) if all elements of S are pairwise comparable (resp. incomparable). The set of all chains (resp. antichains) of P is denoted by $\mathscr{C}(P)$ (resp. $\mathscr{A}(P)$). If all elements of a poset are comparable (i.e. P is a chain) then the poset is said to be a **totally ordered set**.

Notation. From now on, $(P, \leq)$ denotes an arbitrary poset and S is a subset of P.

In general other binary relations are induced from $\leq$:

- The **strict order** of $\leq$, denoted $<$, and given by:

$$(\forall p, q \in P)\ p < q \iff p \leq q \text{ and } p \neq q$$

  The partial order $\leq$ is then the reflexive closure of $<$ (i.e $<^{=} = \leq$).
- The **dual order** of $\leq$, denoted $\geq$, and given by:

$$(\forall p, q \in P)\ p \geq q \iff q \leq p$$

  The poset $(P, \geq)$ is called the **dual poset** of $(P, \leq)$. The **strict dual order** is denoted $>$.

Before going deeper into the order-theoretic notions, we present below some posets:

- **The Powerset Poset.** A basic example of a partial order is the powerset $\wp(E)$ on some set E ordered by set inclusion (i.e. $(\wp(E), \subseteq)$). Indeed, the binary relation $\subseteq$ induces a partial order between subsets of E since it is reflexive (i.e. $(\forall A \in \wp(E))\ A \subseteq A$), transitive ($(\forall A, B, C \in \wp(E))\ A \subseteq B$ and $B \subseteq C$ then $A \subseteq C$ and anti-symmetric (i.e. $(\forall A, B \in \wp(E))$ $A \subseteq B$ and $B \subseteq A$ then $A = B$). It is also important to note that $\subseteq$ does not induce a total order. Indeed, for $E = \{a, b, c\}$, $\{a, b\}$ and $\{b, c\}$ are incomparable.
- **A Poset on Natural Numbers.** The set of natural numbers $\mathbb{N}$ is totally ordered by the $\leq$ relation. One can easily verify that $(\mathbb{N}, \leq)$ is a partial order where every pair of elements are comparable. This poset is clearly infinite.
- **Another Poset on Natural Numbers.** It is clear that for one set, one can create many partial orders on it depending on the purpose. For instance, the set of natural numbers $\mathbb{N}$ can also be partially ordered by the relation *divides* /: e.g. 2 divides 4 but 6 and 10 are incomparable.

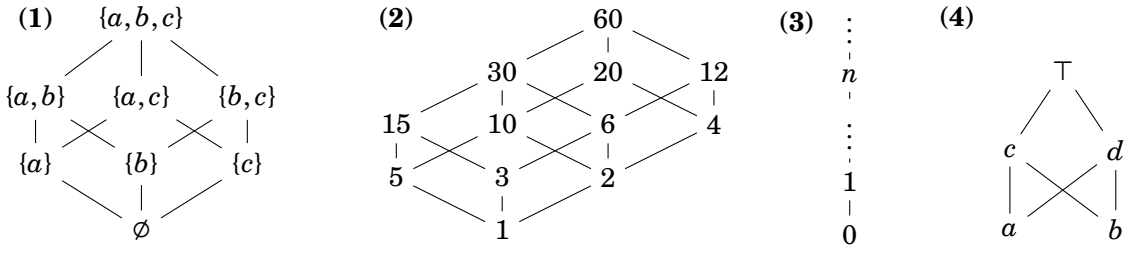


Figure 2.1: From left to right: **(1)** Poset $(\wp(\{a,b,c\}),\subseteq)$. **(2)** Poset of divisors of 60 ordered by $\underline{divides}$. **(3)** The totally ordered set of natural numbers $(\mathbb{N},\leq)$ and **(4)** A Poset built on $\{a,b,c,d,\top\}$ because we are free to think about other posets without giving any meaning to them.

Fig. 2.1 presents a diagrammatic representation of the aforementioned posets. We will also see further in this section why these diagrams, namely **Hasse Diagrams** or **Line Diagrams**, encompass all the information of a (finite) partial order.

We will now dive on some base definitions and operators on partially ordered sets.

Definition 2.3. Let $p,q \in P$, then:

- The **principal ideal** of p is the set of all elements below p: $\downarrow p = \{x \in P \mid x \leq p\}$.
- The **principal filter** of p is the set of all elements above p: $\uparrow p = \{x \in P \mid p \leq x\}$.
- The **interval** $[p,q]$ is the set of elements between p and q: $[p,q] = \uparrow p \cap \downarrow q$

Definition 2.4. Let $p,q \in P$, we say that p is a **direct lower neighbor** of q or q is a **direct upper neighbor** of p and we write $p \prec q$ iff: $[p,q] = \{p,q\}$. In other words, $p < q$ and there is no elements strictly between p and q. The set of direct lower (resp. upper) neighbors of an element $p \in P$ is denoted $lowers(p)$ (resp. $uppers(p)$) and is given by:

$$lowers(p) = \{x \in P \mid x \prec p\} \qquad uppers(p) = \{x \in P \mid p \prec x\}$$

The Hasse Diagram. Any *finite poset* $(P,\leq)$ can be represented by a **Hasse diagram** where:

- Elements of P are represented by small symbols in the plane.
- If $a,b \in P$ and $a \prec b$, the symbol corresponding to a is depicted below the symbol corresponding to b (order is read bottom-up) and a segment of line is drawn between them.

It is clear that the partial order can be deduced from this diagram using reflexivity (i.e. we know that all elements are below themselves) and transitivity (i.e. if $a \prec b \prec ... \prec c$ then $a \leq c$). More formally, the partial order $\leq$ related to the binary relation $\prec$ is then the reflexive and transitive closure of the relation $\prec$, that is: $< = \prec^T$ and $\leq = <^=$. Fig. 2.1 depicts the Hasse diagram of the three aforementioned posets. Consider, for instance the poset $(\wp(\{a,b,c\}),\subseteq)$ depicted in **(1)** (i.e. the powerset of $\{a,b,c\}$ ordered by set inclusion). We have $\emptyset \subseteq \{a\} \subseteq \{a,b\} \subseteq \{a,b,c\}$. The notions presented in the beforehand definitions are illustrated below.

- The principal filter of $\{a\}$ is the set of supersets of $\{a\}$, that is: $\uparrow \{a\} = \{\{a\},\{a,b\},\{a,c\},\{a,b,c\}\}$.
- The principal ideal of $\{a,b\}$ is the set of subsets of $\{a,b\}$, that is: $\downarrow \{a,b\} = \{\emptyset,\{a\},\{b\},\{a,b\}\}$.

- The interval $[\{a\},\{a,b\}]$ is the subsets that are between $\{a\}$ and $\{a,b\}$, that is $[\{a\},\{a,b\}] = \{\{a\},\{a,b\}\}$. Hence, we can say that $\{a\}$ and $\{a,b\}$ are direct neighbors.
- We have $lowers(\{a\}) = \{\emptyset\}$ and $uppers(\{a\}) = \{\{a,b\},\{a,c\}\}$.

Definition 2.5. A set $S \subseteq P$ is said to be a **lower ideal** or a **downset** if:

$$(\forall s \in S, \forall x \in P)\ x \leq s \Rightarrow x \in S$$

Dually, it is said to be an **upper ideal** or an **upset** if:

$$(\forall s \in S, \forall x \in P)\ x \leq s \Rightarrow x \in S$$

The set of all lower (resp. upper) ideals of $(P,\leq)$ is denoted $\mathscr{O}(P)$ (resp. $\mathscr{U}(P)$).

Definition 2.6. The **down closure** (resp. **up closure**) of S, denoted by $\downarrow S$ (resp. $\uparrow S$), is given by the set of elements in P that have at least one element $s \in S$ above (resp. below) it. In other words, it associates the smallest downset (resp. upset) enclosing S and it is given by:

$$\downarrow S = \{x \in P \mid (\exists s \in S)\, x \leq s\} = \bigcup_{s \in S} \downarrow s \qquad \uparrow S = \{x \in P \mid (\exists s \in S)\, s \leq x\} = \bigcup_{s \in S} \uparrow s$$

Note 2.1. One can show that $\mathscr{U}(P) = \{\uparrow S \mid S \subseteq P\}$ and $\mathscr{O}(P) = \{\downarrow S \mid S \subseteq P\}$. □

Example 2.1. Reconsider Fig. 2.1 **(1)**, the set $S = \{\{a\},\{a,b\},\{a,c\}\}$ is not a lower ideal in $(\wp(\{a,b,c\}),\subseteq)$ since $\emptyset \notin S$ but $\emptyset$ is below $\{a\}$. It is also not a upper ideal since $\{a,b,c\} \notin S$. We have $\downarrow S = \{\emptyset,\{a\},\{a,b\},\{a,c\}\}$ and $\uparrow S = \{\{a\},\{a,b\},\{a,c\},\{a,b,c\}\}$ which are respectively lower ideals and upper ideals. □

Definition 2.7. An element $x \in P$ is said to be a **lower bound** (resp. **upper bound**) of S if it is below (resp. above) all elements of S. The set of lower (resp. upper) bounds of S in P, denoted by S^ℓ (resp. S^u), is given by:

$$S^\ell = \{x \in P \mid (\forall s \in S)\, x \leq s\} = \bigcap_{s \in S} \downarrow s \qquad S^u = \{x \in P \mid (\forall s \in S)\, s \leq x\} = \bigcap_{s \in S} \uparrow s$$

Example 2.2. Consider the poset of natural numbers ordered by division $(\mathbb{N},/)$. For $x,y \in \mathbb{N}$, the principal filters $\downarrow x$ and $\downarrow y$ will get respectively the set of their divisors while $\{x,y\}^\ell = \downarrow x \cap \downarrow y$ will get the set of their **common divisors**. For example, the set of divisors of 30 is given by $\downarrow 30 = \{1,2,3,5,6,10,15,30\}$ while the set of divisors of 12 is given by $\downarrow 12 = \{1,2,3,4,6,12\}$. Hence, the set of common divisors of 30 and 12 is given by: $\{12,30\}^\ell = \downarrow 30 \cap \downarrow 12 = \{1,2,3,6\}$. □

Note 2.2. It is worth to notice that, particularly, for the empty set $\emptyset \in \wp(P)$, $\uparrow \emptyset = \downarrow \emptyset = \emptyset$ and $\emptyset^\ell = \emptyset^u = P$. Note also that S^ℓ (resp. S^u) is a lower (resp. upper) ideal in $(P,\leq)$. □

2.2.2 Creating Partial Orders from Pre-orders

We will explore here in this section an interesting transformation that allows one to build an equivalence relation and a partial order from any pre-order. Recall that given a set E, a binary relation $\rightarrow$ on E is said to be a **pre-order** on E iff it is *reflexive* and *transitive*. The pair $(E, \rightarrow)$ is said to be a **pre-ordered set**.

Proposition 2.1. Let the binary relation $\leftrightarrow$ on E be defined as follow:

$$\leftrightarrow := \{(a,b) \in E \times E \mid a \rightarrow b \text{ and } b \rightarrow a\}$$

The relation $\leftrightarrow$ is an equivalence relation on E.

Proof. The proof is trivial for the three properties. ∎

Note 2.3. Recall that the *quotient set* $E \setminus \leftrightarrow$ forms a partition on E. □

Theorem 2.1. Let the binary relation $\leq$ on the quotient set $E \setminus \leftrightarrow$ be defined as follow:

$$\leq := \{(A,B) \in (E \setminus \leftrightarrow) \times (E \setminus \leftrightarrow) \mid (\exists a \in A, \exists b \in B)\, a \rightarrow b\}$$

The pair $(E \setminus \leftrightarrow, \leq)$ is a poset called the **quotient poset** of $\rightarrow$.

Proof. Recall that elements of $E \setminus \leftrightarrow$ are not empty sets. Showing the reflexivity and transitivity of $\leq$ is trivial. Let us show now that $\leq$ is anti-symmetric. Let $A, B \in E \setminus \leftrightarrow$ s.t. $A \leq B$ and $B \leq A$. Hence $\exists a \in A, \exists b \in B$ s.t. $a \rightarrow b$ and $\exists c \in A, \exists d \in B$ s.t. $d \rightarrow c$. By definition of the quotient set we have $a \rightarrow c$, $c \rightarrow a$, $b \rightarrow d$ and $d \rightarrow b$. Hence, $b \rightarrow d \rightarrow c \rightarrow a$. We conclude that $a \leftrightarrow b$ or in other words $A = B$. ∎

Example 2.3. Fig. 2.2 **(left)** presents a pre-ordered set $(E, \rightarrow)$ with $E = \{a,b,c,d,e,f\}$. Notice that reflexivity and transitivity is omitted from the representation of the pre-order, i.e. the pre-order is actually the result of the reflexive and transitive closure of the represented relation. The quotient set of the related equivalence relation is the **strongly connected component** of the directed graph and the quotient poset is represented via its Hasse diagram in Fig. 2.2 **(right)**. □

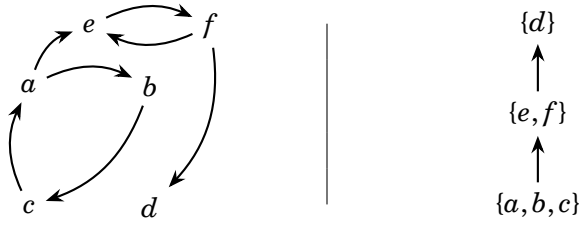


Figure 2.2: A preorder on $\{a,b,c,d,e,f\}$ **(left)** and its quotient poset **(right)**

2.2.3 Minimal, Minimum, Infimum and their Duals

If we consider a subset $S \subseteq P$, informally speaking, not all elements in S or $P \backslash S$ have the same position. Some elements x are in the "interior" of S, that is $\exists y, z \in S$ such that $y < x < z$; while some others are on its "border", that is there is no element in S below (or above) them. These elements in the border are called minimal and maximal elements (see Definition 2.8).

Definition 2.8. An element $s \in S$ is said to be a **minimal** (resp. **maximal**) element in S if all its strict lower (resp. upper) bounds are outside S. The set of minimal (resp. maximal) elements of S is the set denoted by $min(S)$ (resp. $max(S)$) and given by:

$$min(S) = \{s \in S \mid \downarrow s \cap S = \{s\}\} \qquad max(S) = \{s \in S \mid \uparrow s \cap S = \{s\}\}$$

Note 2.4. One should note that for any $S \subseteq P$, minimal (maximal) elements are incomparable. In other words, $min(S)$ and $max(S)$ are antichains in $(P, \leq)$. □

Example 2.4. Consider Fig. 2.1 **(1)**, let be $S = \{\{a\}, \{a,b\}, \{a,c\}\}$. We have $min(S) = \{\{a\}\}$ and $max(S) = \{\{a,b\}, \{a,c\}\}$. □

Before going further on characterizing the elements of a subset S, one important observation is made in Lemma 2.1 on the relationship between minimal elements of a subset S and the minimal elements of its up-closure $\uparrow S$.

Lemma 2.1. We have:

$$min(\uparrow S) = min(S) \qquad max(\downarrow S) = max(S)$$

Proof. We prove by double inclusion the property $min(\uparrow S) = min(S)$:

- **($\subseteq$)** Let $x \in min(\uparrow S)$ and suppose that $x \notin S$. Since $x \in min(\uparrow S)$ we have $x \in \uparrow S$, that is $\exists y \in S$ s.t. $y \leq x$ but $y \neq x$ since $x \notin S$. Thus, $\exists y \in \uparrow S$ s.t. $y \leq x$ but $y \neq x$. Thus $y \in \downarrow x \cap \uparrow S$ with $y \neq x$ which contradicts the fact that $x \in min(\uparrow S)$ (i.e. $\downarrow x \cap \uparrow S = \{x\}$). We conclude that $x \in S$. Suppose now that $x \notin min(S)$, that $\exists y \in S$ s.t. $y \leq x$ and $y \neq x$. Hence, $y \in \uparrow S \cap \downarrow x$ which contradicts the fact that $x \in min(\uparrow S)$. Thus $x \in min(S)$. We conclude that $min(\uparrow S) \subseteq min(S)$.
- **($\supseteq$)** Let $x \in min(S)$, thus $x \in \uparrow S$. Suppose that $x \notin min(\uparrow S)$ that is $\exists y \in \uparrow S$ such that $y \leq x$ and $y \neq x$. Thus $\exists z \in S$ such that $z \leq y \leq x$ with $z \neq x$. Hence, $z \in \downarrow x \cap S$ with $z \neq x$ which contradicts the fact that $x \in min(S)$. We conclude that $x \in min(\uparrow S)$ or more generally $min(S) \subseteq min(\uparrow S)$.

One can follow the same steps to show that $max(\downarrow S) = max(S)$. ▌

We have seen in Example 2.4 that maximal elements of a subset S could be multiple and in such case they are incomparable. On the same example, we have seen that the considered subset S has a unique minimal element which at the same time is below all elements of S, this particular element is said to be a minimum and is formally presented in Definition 2.9.

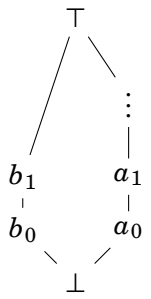


Figure 2.3: there is a breach on the wall for $S = \{\bot, b_0, b_1\} \cup \{a_i \mid i \in \mathbb{N}\}$, i.e. $max(S) = \{b_1\}$.

Definition 2.9. An element $m \in S$ is said to be:

- A **minimum** or a **smallest element** of S if it is below all elements of S. Formally:

$$m \in S \text{ and } (\forall s \in S)\ m \leq s$$

- A **maximum** or a **greatest element** of S if it is above all elements of S. Formally:

$$m \in S \text{ and } (\forall s \in S)\ s \leq m$$

The minimum (resp.maximum) does not necessarily exist. Moreover, if it exists then it is unique. Formally, we say that:

- S **has a minimum** or S is **minimum-handle**[2] if $S^{\ell} \cap S \neq \emptyset$.
- S **has a maximum** or S is **maximum-handle** if $S^{u} \cap S \neq \emptyset$.

Example 2.5. Consider Fig. 2.1 **(1)**, subset $S = \{\{a\}, \{a,b\}, \{a,c\}\}$ is a minimum-handle since $\{a\}$ is a subset of all elements of S. However, S does not have a maximum since elements $\{a,b\}$ and $\{a,c\}$ are incomparable and has no element above them in S (i.e. they are maximal elements). □

Note 2.5. If the poset has a maximum, we call it the **top** element and we denote it $\top$. Dually, if the poset has a minimum, we call it the **bottom** element and we denote it $\bot$. □

It is important to understand the difference between maximal elements and the maximum element. In fact, when a subset S has a maximum m then $max(S) = \{m\}$. In other words, "If S has a maximum then S has a unique maximal element". However, the converse (i.e. "If S has a unique maximal element then S has a maximum") is not true. In fact, this statement holds for finite posets but does not necessarily hold for an arbitrary poset. Consider, for instance the infinite poset $(P, \leq)$ depicted in Fig. 2.3 where $P = \{\bot, \top, b_0, b_1\} \cup \{a_i \mid i \in \mathbb{N}\}$ with:

- $\bot \leq b_0 \leq b_1 \leq \top$.
- $(\forall i \in \mathbb{N})\ \bot \leq a_i$, $a_i \leq a_{i+1}$ and $a_i \leq \top$.

[2]The terms **minimum-handle** and **maximum-handle** used in Definition 2.9 and the terms **minimal-handle** and **maximal-handle** used in Definition 2.10 come from Martinez et al's. [129] paper on multilattices.

Consider, the subset $S = P \setminus \{\top\} = \{\bot, b_0, b_1\} \cup \{a_i \mid i \in \mathbb{N}\}$. It is clear that: $max(S) = \{b_1\}$. That is, S has a single maximal element. Yet, S has no maximum (i.e. is not a maximum-handle) since b_1 is incomparable with elements a_i for all $i \in \mathbb{N}$. In fact, elements a_i has no maximal elements in $max(S)$ above them (i.e. $S \not\subseteq \downarrow max(S)$). Definition 2.10 formalizes the following (intuitive) property (i.e. every element in S has at least one maximal element above it).

Definition 2.10. We say that S is:

- A **minimal-handle** if $\forall s \in S$, $\exists m \in min(S)$ such that $m \leq s$. In other words:

$$S \subseteq \uparrow min(S)$$

- A **maximal-handle** if $\forall s \in S$, $\exists m \in max(S)$ such that $s \leq m$. In other words:

$$S \subseteq \downarrow max(S)$$

Note that for an upper ideal $S \in \mathscr{U}(S)$ (i.e. $S = \uparrow S$), saying that S is minimal-handle is equivalent to say that $S = \uparrow min(S)$. One should note also that in the case where S is minimal-handle and S has a unique minimal element then S is minimum-handle. It is also worthwhile to notice that, trivially, for any poset we have that $\emptyset$ is a minimal-handle and a maximal-handle since $\uparrow \emptyset = \downarrow \emptyset = \emptyset$. However, it is neither minimum-handle nor maximum-handle since the $\emptyset$ is empty by its essence and does not contain any element.

Definition 2.11. We have:

- The largest lower bound of S (i.e. the maximum of S^ℓ) if it exists is called the **infimum** or the **meet** of S and is denoted $inf(S)$ or $\bigwedge S$. Moreover, we have $S^\ell = \downarrow (\bigwedge S)$, that is:

$$(\forall p \in P)\ p \in S^\ell \iff p \leq \bigwedge S$$

- The smallest upper bound of S (i.e. the minimum of S^u) if it exists is called the **supremum** or the **join** of S and is denoted $sup(S)$ or $\bigvee S$. Moreover, we have $S^u = \uparrow (\bigvee S)$, that is:

$$(\forall p \in P)\ p \in S^u \iff \bigvee S \leq p$$

There is a tight relationship between the minimum and the infimum. In fact, it is easy to see that if S has a minimum then the infimum of S is its minimum. In other words, having the minimum is a stronger property than having an infimum. One important remark is the fact that for any $S \subseteq P$, S^u has an infimum if and only if it has a minimum. That is, having a minimum is no longer a stronger property than having an infimum when a set of upper bounds is considered. Lemma 2.2 formalizes and generalizes this observation and its dual.

Lemma 2.2. Let $(P, \leq)$ be a poset, $S \subseteq P$, we have:

- For any $A \subseteq S^\ell$, if A has a join $\bigvee A \in P$ then $\bigvee A \in S^\ell$.
- For any $A \subseteq S^u$, if A has a meet $\bigwedge A \in P$ then $\bigwedge A \in S^u$.

Proof. Let $A \subseteq S^\ell$, we have by definition: $(\forall s \in S\ \forall a \in A)\ a \leq s$, that is $S \subseteq A^u$. Since $\bigvee A$ is the least upper bound of A and all elements of S are upper bounds of A then: $(\forall s \in S)\ \bigvee A \leq s$. We conclude that $\bigvee A \in S^\ell$. Same steps can be followed to show the second part of the Lemma. ∎

Proposition 2.2. In case of existence, we have:

$$\bigwedge S = \bigvee S^\ell \qquad\qquad \bigvee S = \bigwedge S^u$$

Proof. Suppose that S has its infimum $\bigwedge S$. By definition, $\bigwedge S$ is the maximum of S^ℓ. Let us show now that this maximum is the join of S^ℓ that is the minimum of $\left(S^\ell\right)^u$. Since $\bigwedge S$ is the maximum of S^ℓ then $\bigwedge S$ is an upper bound of S^ℓ (i.e. $\bigwedge S \in \bigwedge \left(S^\ell\right)^u$). Moreover, let $u \in \left(S^\ell\right)^u$, then $(\forall x \in S^\ell)\ x \leq u$. Hence, since $\bigwedge S \in S^\ell$, we have for all $u \in \left(S^\ell\right)^u$: $\bigwedge S \leq u$. Since $\bigwedge S \in \bigwedge \left(S^\ell\right)^u$ then $\bigwedge S$ is the minimum of $\left(S^\ell\right)^u$ (i.e. $\bigwedge S = \bigvee S^u$). One can prove the other property dually. ∎

Note 2.6. According to Proposition 2.2 and since for a poset $(P, \leq)$ we have $\emptyset^\ell = \emptyset^u = P$, then the empty set has a meet (resp. join) iff the poset is upper-bounded (resp. lower bounded):

$$\bigwedge \emptyset = \bigvee P = \top \qquad\qquad \bigvee \emptyset = \bigwedge P = \bot.$$

□

2.2.4 Summary

Table 2.1 resumes the different notations and properties on subsets we have learned so far.

Notation	**Meaning with $(P,\leq)$ a poset, $S \subseteq P$ and $p,q \in P$**
$\wp(P)$	Powerset of P: $\wp(P) = \{A \mid A \subseteq P\}$
$\mathscr{C}(P)$	Set of chains on $(P,\leq)$
$\mathscr{A}(P)$	Set of antichains on $(P,\leq)$
$\uparrow p$	Principal filter of p : $\uparrow p = \{x \in P \mid p \leq x\}$
$\downarrow p$	Principal ideal of p : $\downarrow p = \{x \in P \mid x \leq p\}$
$[p,q]$	Interval $[p,q] = \uparrow p \cap \downarrow q$
$p \prec q$	p is a lower neighbor of q that is $[p,q] = \{p,q\}$
$lowers(p)$	lower neighbors of p: $lowers(p) = \{x \in P \mid x \prec p\} = max(\downarrow p \setminus \{p\})$
$uppers(p)$	upper neighbors of p: $uppers(p) = \{x \in P \mid p \prec x\} = min(\uparrow p \setminus \{p\})$
$min(S)$	Set of minimal elements of S : $min(S) = \{s \in S \mid \downarrow s \cap S = \{s\}\}$
$max(S)$	Set of maximal elements of S : $max(S) = \{s \in S \mid \uparrow s \cap S = \{s\}\}$
$\uparrow S$	Up-closure $\uparrow S = \bigcup_{s \in S} \uparrow s$
$\downarrow S$	Down-closure $\downarrow S = \bigcup_{s \in S} \downarrow s$
$\mathscr{U}(P)$	Set of upper ideals (upsets) : $\mathscr{U}(P) = \{\uparrow S \mid S \subseteq P\}$
$\mathscr{O}(P)$	Set of lower ideals (downsets) : $\mathscr{O}(P) = \{\downarrow S \mid S \subseteq P\}$
S^u	Set of upper bounds of S : $S^u = \bigcap_{s \in S} \uparrow s$
S^ℓ	Set of lower bounds of S : $S^\ell = \bigcap_{s \in S} \downarrow s$
Property	**Meaning**
S is maximum-handle	S has a maximum $m \in S$, that is: $S \subseteq \downarrow m$
S is minimum-handle	S has a minimum $m \in S$, that is: $S \subseteq \uparrow m$
S is maximal-handle	$S \subseteq \downarrow max(S)$
S is minimal-handle	$S \subseteq \uparrow min(S)$
S has an infimum	S^ℓ is maximum-handle and this maximum is denoted $\bigwedge S$
S has a supremum	S^u is minimum-handle and this minimum is denoted $\bigvee S$

Table 2.1: Base Order-theoretic notations and properties

2.3 Partially Ordered Sets Properties

We explore in this section some properties of posets. The classification relies in general on the properties that some subsets can have with regard to the existence of their maximum, minimum, maximal elements, minimal elements, infimum or supremum.

2.3.1 Bounded Posets

A poset $(P,\leq)$ is said to be:

- **Upper-bounded** if it has a top element $\top \in P$ such that:

$$(\forall p \in P)\ p \leq \top$$

- **Lower-bounded** if it has a bottom element $\bot \in P$ such that:

$$(\forall p \in P)\ p \geq \bot$$

- **Bounded** if it is both upper-bounded and lower-bounded.

It is clear that not all posets are lower-bounded. In fact, even finite posets can suffer from the lack of the existence of a bottom element. For instance, Fig. 2.1 **(4)** depicts a finite poset that is non lower bounded. If a poset $(P,\leq)$ is not lower-bounded, one can create simply an associated lower-bounded poset $(P_\bot,\leq)$ by adding a bottom element $\bot$ to P (i.e. $P_\bot = P \cup \{\bot\}$. This operation is called **lifting**. A **dual lifting** is the operator of creating a upper-bounded poset $(P^\top,\leq)$ starting from a non upper-bounded poset $(P,\leq)$.

For a bounded poset, an element $p \in P$ is said to be an **atom** (resp. **coatom**) if p is a direct upper (resp. lower) neighbor of the bottom (resp. top) element. Formally, the atoms and the coatoms of a bounded poset are given by:

$$atoms(P) = uppers(\bot) = min(P \backslash \{\bot\}) \qquad coatoms(P) = lowers(\top) = max(P \backslash \{\top\})$$

Example 2.6. For any set E, the poset $(\wp(E),\subseteq)$ is a bounded poset where the bottom element is $\emptyset$ and the top element is E. The atoms are the singleton elements (i.e. $atoms(E) = \{\{e\} \mid e \in E\}$) and the coatoms are given by $coatoms(E) = \{E \backslash \{e\} \mid e \in E\}$.

Interestingly, the poset of natural numbers ordered by division $(\mathbb{N},/)$ is also bounded, the bottom element is 1 and the top element is 0 (if we allow 0/0). The atoms of this infinite poset are the prime numbers while there is no coatoms. □

Note 2.7. Please note that:

- If $(P,\leq)$ is lower-bounded, and by recalling that $\emptyset^u = P$ by definition, then:

$$\bigvee \emptyset = \bot$$

- If $(P,\leq)$ is upper-bounded, and by recalling that $\emptyset^\ell = P$ by definition, then:

$$\bigwedge \emptyset = \top$$

□

2.3.2 Chain-Finite Posets

A Bounded poset requires that the entire poset has both a maximum and a minimum or, equivalently, the empty set has both a supremum and an infimum. Other class of posets require other subsets of the poset to have their maximum or minimum.

A poset is said to be **well-founded**[3] or has the **minimal condition** if all its subsets are minimal-handles (cf. Definition 2.10). An equivalent statement to the minimal condition is that the poset does not have infinite descending chains. That is, all chains in the poset have their minimum. This condition is called the **Descending Chain Condition (DCC)**.

Dually, a poset is said to be **dually well-founded** or has the **maximal condition** if all its subsets are maximal-handles (cf. Definition 2.10). Again, this latter statement is equivalent to say that the poset does not have infinite ascending chains. That is, all chains in the poset has their maximum. This condition is called the **Ascending Chain Condition (ACC)**.

A poset is then said to be **chain-finite** if it has both **ACC** and **DCC** condition, that is all chains of the poset are finite. Analogically, a poset is said to be **antichain-finite** if all its antichains are finite. Clearly, one can show that a poset is **finite** if and only if it is both chain-finite and antichain-finite.

Example 2.7. Consider the poset depicted in Fig. 2.3. It is clear that the poset is **DCC**. However, it does not has the **ACC** since it does have an infinitely ascending chain $a_0 < a_1 < ... < a_n < ....$ □

2.3.3 Chain-Complete Posets

We have seen in Section 2.2.3 that if a subset has a maximum then it has also a supremum. However, requiring the existence of the maximum is much stronger than requiring the existence of the supremum. Here lie a new structure of posets called chain-complete ones.

Definition 2.12. A poset $(P, \leq)$ is said to be:

- **Chain-complete** if all chains in P, including $\emptyset$, has its supremum.
- **Dually chain-complete** if all chains in P, including $\emptyset$, has its infimum.
- **Doubly chain-complete** if it is both chain-complete and dually chain-complete.

Note 2.8. It is clear that all chain-complete posets are lower-bounded since it is required that the empty set has its supremum which is the bottom element (i.e. $\bigvee \emptyset = \bot$). However, a chain-complete poset does not necessarily have a top element. Dual observation can be made about dually chain-complete posets. Hence, it is worthwhile to notice that finite posets are not necessarily chain-complete because of the (hated) empty set. A chain-finite poset is then doubly chain-complete if and only if it is bounded. More generally, posets having the **ACC** are chain-complete if and only if they have a bottom element. Dually, posets having the **DCC** are dually chain-complete if and only if they have a top element. □

[3] A totally ordered set that is well-founded is said to be **well-ordered**

Example 2.8. We have:

- The finite poset depicted in Fig. 2.1 **(4)** is dually chain-complete but not chain-complete due to the lack of the bottom element.
- The poset depicted in Fig. 2.3 is doubly chain-complete even if it is not **ACC**. Indeed, the infinitely ascending chain $a_0 < a_1 < ...$ has a supremum which is the top element $\top$.
- The totally ordered set of real numbers $(\mathbb{R}, \leq)$ is not bounded hence not chain-complete. However, the extended real line $(\overline{\mathbb{R}}, \leq)$ (i.e. $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, \infty\}$) is doubly chain-complete.
- Consider the extended line of rational numbers $(\overline{\mathbb{Q}}, \leq)$ with $\overline{\mathbb{Q}} = \mathbb{Q} \cup \{-\infty, \infty\}$ is neither chain-complete nor dually chain-complete. Consider for instance subsets A and B where:

  $$A = \{q \in \overline{\mathbb{Q}} \mid q^2 < 2 \text{ or } q \leq 0\} \qquad B = \overline{\mathbb{Q}} \backslash A = \{q \in \overline{\mathbb{Q}} \mid q^2 \geq 2 \text{ and } q > 0\}$$

  Recall the positive element $q \in \overline{\mathbb{Q}}$ such that $q^2 = 2$ does not exist (i.e. $\sqrt{2}$ is irrational). Hence, one can write:

  $$A = \{q \in \overline{\mathbb{Q}} \mid q^2 < 2 \text{ or } q \leq 0\} \qquad B = \overline{\mathbb{Q}} \backslash A = \{q \in \overline{\mathbb{Q}} \mid q^2 > 2 \text{ and } q > 0\}$$

  Therefore, A does not have a maximum and B does not have a minimum. Moreover, $A^u = B$ and $B^\ell = A$. Hence, the chain A does not have a supremum and the chain B does not have an infimum. This makes $(\overline{\mathbb{Q}}, \leq)$ not (dually) chain complete. It is clear that $\{A, B\}$ form a partition on $\overline{\mathbb{Q}}$. In fact, the pair (A, B) is said to be a **Dedekind's Cut**. Interestingly, this cut characterizes uniquely the irrational number $\sqrt{2}$ separating A and B. □

2.3.4 Lattices and Complete Lattices

We will now consider a very important structure in this manuscript which is the Lattice structure.

Definition 2.13. A poset $(P, \leq)$ is said to be:

- A **meet-semilattice** if for all nonempty finite subsets $S \subseteq P$, S has a meet (infimum).
- A **join-semilattice** if for all nonempty finite subsets $S \subseteq P$, S has a join (supremum).
- A **lattice** if it is both meet-semilattice and join-semilattice.
- A **complete lattice** if for all subsets $S \subseteq P$ including $\emptyset$, S has its meet and join.

Example 2.9. Below some examples of lattices and non-lattices:

- For any (finite or infinite) set E, the poset $(\wp(E), \subseteq)$ is a complete lattice and it is called the **powerset lattice**. The meet is the set intersection while the join is the set union. For instance, Fig. 2.1 **(1)** depicts the Hasse diagram of the powerset lattice $(\wp(\{a, b, c\}), \subseteq)$. We have $\{\{a, c\}, \{a, b\}\}^\ell = \{\emptyset, \{a\}\}$ and $\{\{a, c\}, \{a, b\}\}^u = \{\{a, b, c\}\}$. Therefore $\bigwedge\{\{a, c\}, \{a, b\}\} = \{a\}$ while $\bigvee\{\{a, c\}, \{a, b\}\} = \{a, b, c\}$.
- The set of natural numbers ordered by division relation $(\mathbb{N}, /)$ is a lattice where the meet is the greatest common divisor **gcd** and the join is the least common multiple **lcm**. Fig. 2.1 depicts the Hasse Diagram of a portion of this lattice. For instance the meet of $\{12, 30\}$ is

gcd$(12,30) = 6$ (i.e. recall that the common divisors are $\{12,30\}^{\ell} = \downarrow 12 \cap \downarrow 30 = \{1,2,3,\mathbf{6}\}$). In fact this lattice is a complete lattice which top element is 0 and bottom element is 1.

- The poset depicted in Fig. 2.1 **(4)** is neither a meet-semilattice nor a join-semilattice. Indeed, we have $\{a,b\}^{u} = \{c,d,\top\}$ which minimal elements are $\{c,d\}$ (i.e. $\{a,b\}^{u}$ has no minimum). Thus, $\{a,b\}$ does not have a join. Moreover, we have $\{c,d\}^{\ell} = \{a,b\}$ which is an antichain. Therefore, $\{c,d\}$ does not have a meet. □

Note 2.9. We have:

- All totally ordered sets $(P,\leq)$ are lattices. Indeed, for any nonempty finite set (i.e. chain) $S \subseteq P$, the infimum is the minimum element while the supremum is the maximum one.
- Since complete lattices require that any subset of P has a meet and join, complete lattices are by definition bounded.
- Finite lattices are trivially complete lattices. □

We will now explore some important properties that (complete) lattices have.

2.3.4.1 Pairs as building blocks

There is a weaker, yet equivalent, definition of meet-semilattices and join-semilattices. In fact, we have the following equivalences:

$$(P,\leq) \text{ is a meet-semilattice} \iff (\forall p,q \in P)\ \{p,q\} \text{ has a meet}$$
$$(P,\leq) \text{ is a join-semilattice} \iff (\forall p,q \in P)\ \{p,q\} \text{ has a join}$$

It is clear that $(P,\leq)$ is a meet-semilattice then all pairs of elements have a meet since pairs are nonempty finite sets. For the converse, one can build the infimum of a nonempty finite set S using pairs of elements as follow:

- If $S = \{p_1\}$ then its infimum is p_1 since $S^{\ell} = \downarrow p_1$.
- If $S = \{p_1,p_2\}$ then the infimum exists under the hypothesis.
- For larger finite sets $\{p_1,p_2,...,p_n\}$, one can show that:

$$\bigwedge\{p_1,p_2,...,p_n\} = \left\{\bigwedge\{p_1,p_2\},p_3,...,p_n\right\}$$

This operation is iterated until obtaining a pair of element which meet exists.

Therefore, rather than checking for all nonempty finite subsets $S \subseteq P$ if S has a meet and a join, one should check the existence of the meet and the join of only pair of elements to prove that the considered poset is a lattice.

From an *algebraic point of view*, one can create two binary operators $\wedge$ and $\vee$ on a lattice based on the meet and the join:

$$(\forall p,q \in P)\ \ p \wedge q := \bigwedge\{p,q\} \text{ and } p \vee q := \bigvee\{p,q\}$$

These two operators are associative, commutative and idempotent.

2.3.4.2 Complete semilattices are complete lattices

Again, there is a weaker, yet equivalent, definition of complete lattices: A poset is a complete lattice if and only if all its subsets have a meet. Indeed, recall that (cf. proposition 2.2), in case of existence of $\bigvee S$, we have:

$$\bigvee S = \bigwedge S^u \tag{2.1}$$

Hence, supposing that all subsets have a meet, we obtain that they have also a join too. It is clear that the dual property is also true.

Note 2.10. After seeing equation 2.1, one could be tempted to say that all finite meet-semilattices are finite lattices. This statement is false since, even if for all S the set S^u is finite, the set S^u could be empty and thus there is no guarantee that it has its meet. Henceforth, we have the two following observations:

- A finite meet-semilattice is a lattice if and only if it is upper-bounded.
- A finite join-semilattice is a lattice if and only if it is lower-bounded. □

2.3.4.3 Complete lattices and chain-completeness

It is clear that complete lattices are (doubly) chain-complete by definition. Theorem 2.2 presents a stronger statement.

Theorem 2.2 (Theorem 3.24, page 68 in [148]). A lattice is complete if and only if it is chain-complete.

2.3.4.4 Suborders of sublattices

For any poset $(P,\leq)$ and any subset $S \subseteq P$, one can build a subposet $(S,\leq)$ (i.e. a restriction of $\leq$ onto S). However, subposets do not necessarily preserve the properties of its parent poset. Definition 2.14 presents important subposets that preserve some properties of their parent complete lattices.

Definition 2.14. Let $(P,\leq)$ be a complete lattice with $\bigwedge_P$ and $\bigvee_P$ denote respectively the meet and the join. Let $S \subseteq P$ be a subset. Subposet $(S,\leq)$ is said to be a:

- **Closure system** or a **meet-structure** on $(P,\leq)$ if: $(\forall A \subseteq S)\ \bigwedge_P A \in S$. Subposet $(S,\leq)$ is then a complete lattice where the meet $\bigwedge_S = \bigwedge_P$. Note that $\top_P \in S$.
- **Kernel system** or a **join-structure** on $(P,\leq)$ if: $(\forall A \subseteq S)\ \bigvee_P A \in S$. Subposet $(S,\leq)$ is then a complete lattice where the meet $\bigvee_S = \bigvee_P$. Note that $\bot_P \in S$.
- **Sublattice** of $(P,\leq)$ if for all nonempty and finite $A \subseteq S$ we have $\bigwedge_P A \in S$ and $\bigvee_P A \in S$.
- **Complete sublattice** of $(P,\leq)$ if it is both closure and kernel system.

Example 2.10. Fig. 2.4 depicts examples of subposets of the powerset lattice $(\wp(\{a,b,c,d\}),\subseteq)$. Subposet **(1)** is neither a meet-semilattice nore a join-semilattice. Subposet **(2)** is a subposet of

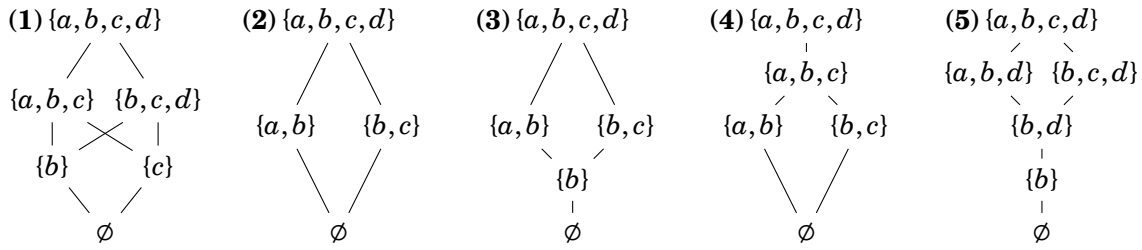


Figure 2.4: From left to right: **(1)** A subposet of $(\wp(\{a,b,c,d\}),\subseteq)$ that is neither a meet-semilattice nor a join-semilattice. **(2)** A subposet of $(\wp(\{a,b,c,d\}),\subseteq)$ that is a (complete) lattice but neither a closure system not a kernel system. **(3)** A closure system of $(\wp(\{a,b,c,d\}),\subseteq)$. **(4)** A kernel system of $(\wp(\{a,b,c,d\}),\subseteq)$. **(5)** A complete sublattice of $(\wp(\{a,b,c,d\}),\subseteq)$.

powerset lattice $(\wp(\{a,b,c,d\}),\subseteq)$ that is complete lattice. However, the meet and the join are not preserved. Hence, it is neither a closure system nor a kernel system on $(\wp(\{a,b,c,d\}),\subseteq)$. Subposets depicted in **(3)** and **(4)** are respectively closure and kernel systems on $(\wp(\{a,b,c,d\}),\subseteq)$ but are not complete sublattices. Subposet **(5)** is clearly a complete sublattice since it does preserve both set intersection and set union. □

Example 2.11. For any poset $(P,\leq)$:

- The poset of upper ideals $(\mathscr{U}(P),\subseteq)$ and the poset of lower ideals $(\mathscr{O}(P),\subseteq)$ are complete sublattices of the complete lattice $(\wp(P),\subseteq)$.
- The poset $(DM(P),\subseteq)$ where $DM(P) := \{S^\ell \mid S \subseteq P\}$ is a closure system on $(\wp(P),\subseteq)$ but not necessarily a kernel system. The notation DM comes from **Dedekind-MacNeille** and it is related to an important notion presented in Section 2.4.4.1. □

Note 2.11. One should note that a sublattice could be a complete lattice but still not a complete sublattice. For instance, consider a set E and an arbitrary proper subset $S \subsetneq E$. It is clear that $(\wp(S),\subseteq)$ is a sublattice of $(\wp(E),\subseteq)$. Moreover, since $(\wp(S),\subseteq)$ is the powerset lattice it is a complete lattice. However, this sublattice is still not a complete sublattice since the top element of $(\wp(S),\subseteq)$ is not in $(\wp(E),\subseteq)$ (i.e. the meet of the emptyset is not preserved). In other words, $(\wp(S),\subseteq)$ is just a kernel system but not a closure system because of the empty set. □

2.3.4.5 Other properties on lattices

Some lattices have more properties than others with regard to their meet and their join. We will not explore here exhaustively the different properties of lattices but just give a glimpse of some notions that we will be using in this dissertation.

Definition 2.15. Let $(P,\leq)$ be a complete lattice, a subset $D \subseteq P$ is said to be $\wedge$**-dense** (i.e. **meet-dense**) iff any $p \in P$ can be seen as a meet of some $S \subseteq D$. Formally:

$$(\forall p \in P\ \exists S \subseteq D)\ p = \bigwedge S.$$

The $\vee$**-denseness** is defined dually.

A meet-dense subset of a complete lattice is then a subset from which one can rebuild the lattice based only on its element and the meet. However, what is the smallest subset dense in $(P,\leq)$. To define this smallest subset, we need to define the notion of irreducibility.

Definition 2.16. Let $(P,\leq)$ be a complete lattice, we say that $p \in P$ is:

- $\wedge$**-irreducible** if it can not be represented as the infimum of strictly larger elements. Formally: $p \neq \bigwedge\{x \in P \mid p < x\}$. The set of all $\wedge$-irreducible elements is denoted $\mathbf{M}(P)$.
- $\vee$**-irreducible** if it can not be represented as the supremum of strictly smaller elements. Formally: $p \neq \bigvee\{x \in P \mid x < p\}$. The set of all $\vee$-irreducible elements is denoted $\mathbf{J}(P)$.

In fact, meet-irreducible elements are the elements that must appear in a dense subset since someone cannot obtain them using the meet of other elements. Formally, for any complete lattice $(P,\leq)$ and for $D \subseteq P$:

$$D \text{ is } \wedge\text{-dense} \Rightarrow \mathbf{M}(P) \subseteq D \qquad\qquad D \text{ is } \vee\text{-dense} \Rightarrow \mathbf{J}(P) \subseteq D$$

Obviously, the set of coatoms of a complete lattice (i.e. lower neighbors of the top elements) are meet-irreducible since the only element that strictly larger than them is the top element. Formally, we have for any complete lattice $(P,\leq)$:

$$coatoms(P) \subseteq \mathbf{M}(P) \qquad\qquad atoms(P) \subseteq \mathbf{J}(P)$$

Theorem 2.3 show a strong link between denseness and irreducibility in finite lattices.

Theorem 2.3 (related to Proposition 12 in [80]). Let $(P,\leq)$ be a finite lattice, then $\mathbf{M}(P)$ and $\mathbf{J}(P)$ are respectively the smallest $\wedge$-dense and $\vee$-dense subsets in P.

Last but not least, a complete lattice is said to be **atomistic** if $atoms(P)$ is join-dense. Dually, it is said to be **coatomistic** if $atoms(P)$ is meet-dense. Note that, we have $\mathbf{J}(P) = atoms(P)$ in an atomistic lattice and $\mathbf{M}(P) = coatoms(P)$ in a coatomistic one.

Example 2.12. We have:

- For any arbitrary subset E, the powerset lattice $(\wp(E),\subseteq)$ is atomistic. Indeed, any subset can be built using the union (i.e. join) of singleton sets (i.e. atoms). The powerset lattice $(\wp(E),\subseteq)$ is also coatomistic.
- In a totally ordered set $(T,\leq)$, we have $\mathbf{M}(T) = \mathbf{J}(T) = T$.
- Consider the complete lattice of natural numbers ordered by division $(\mathbb{N},\leq)$. It is clear that the lattice is not atomistic since 4 is join-irreducible (i.e. it has a unique lower neighbor). In fact, one can verify that the set of join-dense elements is given by:

  $$\mathbf{J}(\mathbb{N}) = \left\{p^i \mid i \in \mathbb{N}^* \text{ and } p \in atoms(\mathbb{N})\right\}$$

  In fact, this is directly linked with the fact that any natural number has a (unique) factorization using the prime numbers (i.e. the atoms here). □

Other particular lattices are studied in the literature, we draw here the reader attention to to distributive lattices.

Definition 2.17. Let $(P,\leq)$ be a lattice. A lattice is said to be **distributive** iff the associated infimum $\wedge$ and supremum $\vee$ binary operator verify the following properties $\forall p,q,w \in P$:

$$p \wedge (q \vee w) = (p \wedge q) \vee (p \wedge w) \text{ and } p \vee (q \wedge w) = (p \vee q) \wedge (p \vee w)$$

Note 2.12. An example of a usual distributive lattice is the powerset lattice. Other variant of lattices exists such as semi-distributive lattices and completely distributive lattices (see [80, 148]). Note that if $(Q,\leq)$ is a sublattice of some distributive lattice $(P,\leq)$ then $(Q,\leq)$ is also distributive. □

2.3.5 Multilattices and Complete Multilattices

The term **multilattice** was introduced for the first time by Benado in [22]. This notion has not received much interest for a long period, but has been unearthed and revisited in the beginning of the 21st century [50, 128, 129] for other purpose. While this may seem illogical, we will start by presenting multilattices following [50, 128, 129]. We will then understand the main difference between Benado's multilattices [22] and Martinez's multilattices [50, 128, 129] afterward.

Multilattices, as their names imply, are related in their definition with lattices. Simply put, multilattices are a relaxation of lattices where rather than demanding that the set of lower (resp. upper) bounds of each nonempty finite subset is minimum-handle (resp. maximum-handle), multilattices demand that the set of lower (resp. upper) bounds of each nonempty finite subset is minimal-handle (resp. maximal-handle).

Definition 2.18. A poset $(P,\leq)$ is said to be:

- A **meet-multisemilattice** if for all nonempty finite $S \subseteq P$ we have S^ℓ is maximal-handle, that is:

  (M) $$S^\ell = \downarrow max\left(S^\ell\right)$$

  The set $max(S^\ell)$ is called the **multi-infimum** of S and is denoted minf(S).
- A **complete meet-multisemilattice** if condition (M) holds for all $S \subseteq P$.
- A **join-multisemilattice** if for all non empty subsets $S \subseteq P$ we have S^u is minimal-handle, that is:

  (J) $$S^u = \uparrow min\left(S^u\right)$$

  The set $min(S^u)$ is called the **multi-supremum** of S and is denoted msup(S).
- A **complete join-multisemilattice** if condition (J) holds for all $S \subseteq P$.

Note that when **property (M)** (resp. **property (J)**) holds for some subset $S \subseteq P$, we will say that S **has all its multi-infima** (resp. S **has all its multi-suprema**).

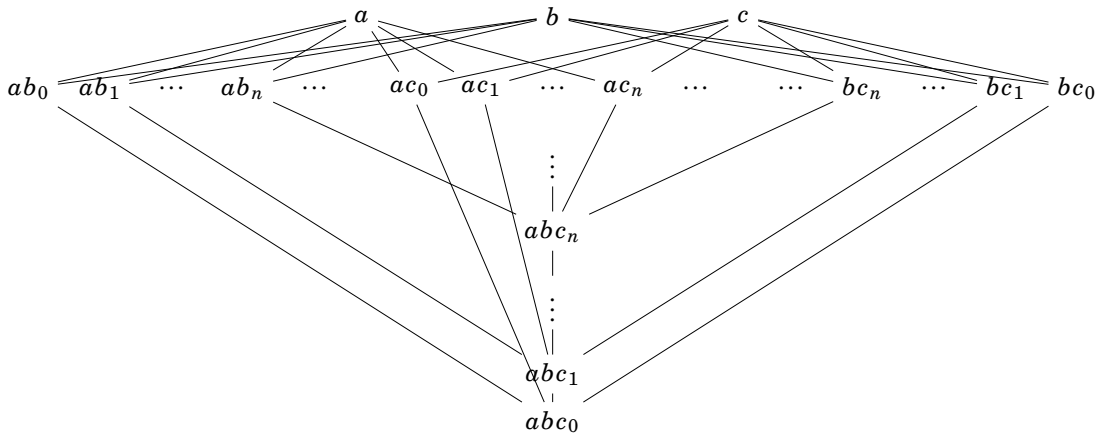


Figure 2.5: For all x, y in this poset, $\{x, y\}$ has all its multi-infima. That is, this poset is a multistructure following [22], however it is not a multilattice following definition 2.18.

Note 2.13. It is clear that all finite posets, or more generally chain-finite posets, are complete multilattices. Hence, conversely to lattices, the notion of multilattices has no importance when finite posets are considered. One should also note that all lattices are multilattices and all complete lattices are complete multilattices. □

Example 2.13. Rather than giving an example about a multilattice, we show here that the property of being a multilattice does not trivially hold. Consider for instance the poset $(P, \leq)$ depicted in Fig. 2.6 where $P = \{c_i \mid i \in \mathbb{N}\} \cup \{a, b\}$ s.t. $(\forall i \in \mathbb{N})\ c_i \leq c_{i+1}$, $c_i \leq a$ and $c_i \leq b$. It is clear that $\{a, b\}^\ell = \{c_i \mid i \in \mathbb{N}\}$. Moreover, $max(\{c_i \mid i \in \mathbb{N}\}) = \emptyset$. Hence, $\{a, b\}^\ell \neq \downarrow max(\{a, b\}^\ell)$. Therefore, $(P, \leq)$ is not a meet-multisemilattice. □

In the following of this section, we revisit some important differences between multilattices and lattices. We will try to answer the three properties discussed on lattices:

1. Are pairs the building blocks of a multilattice?
2. Are complete meet-multisemilattice complete multilattices?
3. What is the relationship between complete multilattices and chain-completeness?

2.3.5.1 The pairs are no longer the building blocks

Recall that the pairs are the building blocks of lattices as we have seen in section 2.3.4.1. But does this property remain for multilattices? Let us start by defining the two following conditions for a given poset $(P, \leq)$:

(MM2) $$(\forall x, y \in P)\ \{x, y\}^\ell = \downarrow max\left(\{x, y\}^\ell\right)$$

(MJ2) $$(\forall x, y \in P)\ \{x, y\}^u = \uparrow min\left(\{x, y\}^u\right)$$

In fact, Benado defined multilattices in the seminal paper [22] as posets for which condition (MM2) and condition (MJ2) hold. Such posets will be called **Benado's multilattices** in this paper.

It is clear that all multilattices following definition 2.18 are Benado's multilattices. However, do we have an equivalence? The answer is negative as shown in [129] and in the following counter-example.

Example 2.14. Let be the poset $(P,\leq)$ depicted in Fig. 2.5 where $P = \{abc_i \mid i \in \mathbb{N}\} \cup \{ab_i \mid i \in \mathbb{N}\} \cup \{ac_i \mid i \in \mathbb{N}\} \cup \{bc_i \mid i \in \mathbb{N}\} \cup \{a,b,c\}$ and:

- $(\forall i \in \mathbb{N})\ abc_i \leq ab_i$, $abc_i \leq ac_i$ and $abc_i \leq bc_i$.
- $(\forall i \in \mathbb{N})\ ab_i \leq a$ and $ab_i \leq b$.
- $(\forall i \in \mathbb{N})\ ac_i \leq a$ and $ac_i \leq c$.
- $(\forall i \in \mathbb{N})\ bc_i \leq b$ and $bc_i \leq c$.

One could verify that this poset is a Benado's multilattice (i.e. both (MM2) and (MJ2) conditions hold), however, this poset is still not a multilattice following definition 2.18. Indeed, considering the non empty finite set $\{a,b,c\}$, we have $\{a,b,c\}^\ell = \{abc_i \mid i \in \mathbb{N}\}$ while $max(\{a,b,c\}^\ell) = \emptyset$. It follows that $\{a,b,c\}^\ell \neq\downarrow max(\{a,b,c\}^\ell)$. □

Hence, the pairs are no longer the building blocks of a multilattice (i.e. being a multilattice following Definition 2.18 is a more restrictive property than being a Benado's multilattice).

2.3.5.2 No more "Buy one, get one for free"

Again, another adage for lattices $(P,\leq)$ is the "Buy one, get one for free". Indeed, if all subsets in P have their meet then all subsets have also their joins. In other words, complete semilattices are complete lattices. However, this adage does no longer hold for multilattices. In fact, a complete join-multisemilattice could be not a meet-multisemilattice. Indeed, Fig. 2.6 presents a complete join-multisemilattice since it has the Descending Chain Condition. Yet, it is not a meet-multisemilattice (i.e. $\{a,b\}^\ell = \{c_i \mid i \in \mathbb{N}\}$ with $max(\{a,b\}^\ell) = \emptyset$). It follows that the *"buy one, get one for free"* adage does no longer hold for complete multilattices.

2.3.5.3 Chain-complete posets are complete meet-multisemilattice

We have seen that all chain-complete lattices are complete lattices and *vice-versa*. What is then the relationship between complete multilattices and chain-complete posets? Theorem 2.5 gives an answer to this question. However, the proof of this theorem requires the usage of the so called Zorn's Lemma[4].

Definition 2.19 (Zorn's Lemma). Let $(P,\leq)$ be a poset, if every chain in $(P,\leq)$ has an upper-bound, then $(P,\leq)$ has a maximal element. Formally:

$$(\forall C \in \mathscr{C}(P))\ C^u \neq \emptyset \implies max(P) \neq \emptyset$$

A stronger statement, yet equivalent, of Zorn's Lemma is stated in the following theorem:

[4] I am grateful to JOZEF PÓCS for attracting my attention to Zorn's Lemma.

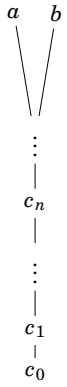


Figure 2.6: A complete join-multisemilattice but not a meet-multisemilattice

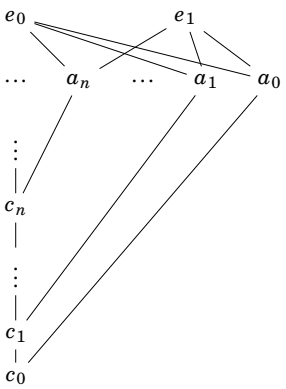


Figure 2.7: A complete multilattice that is not chain-complete

Theorem 2.4 (Zorn's Lemma II). Let $(P,\leq)$ be a poset, we have:

$$(\forall C \in \mathscr{C}(P))\ C^u \neq \emptyset \implies P = \downarrow max(P)$$

Zorn's Lemma need to be considered as an **axiom** since it is equivalent to **axiom of choice (AC)** (see Definition 2.1).

Theorem 2.5. Under *Axiom of Choice* ***(AC)*** assumption, we have:

- All chain-complete posets are complete meet-multisemilattice.
- All dually chain-complete posets are complete join-multisemilattice.
- All doubly chain-complete posets are complete multilattices.

Proof. We show here the first statement of the theorem. Let $(P,\leq)$ be a chain-complete poset and let $S \subseteq P$. We show here that $S^\ell = \downarrow max(S^\ell)$. It is straightforward by definition and independently from any assumption that $\downarrow max(S^\ell) \subseteq S^\ell$. It remains to show that $S^\ell \subseteq \downarrow max(S^\ell)$. Since $(P,\leq)$ is *chain-complete*, then every $C \subseteq S^\ell$ has its join $\bigvee C \in P$. Hence, according to Lemma 2.2 and since $C \subseteq S^\ell$ then $\bigvee C \in S^\ell$. Thus every chain C in the sub-poset $(S^\ell,\leq)$ has an upper bound $\bigvee C \in S^\ell$. According to *Zorn's Lemma*, and by recalling that the Axiom of choice is equivalent to Zorn's Lemma , we have $S^\ell = \downarrow max(S^\ell)$. Hence, $(P,\leq)$ is a complete meet-multisemilattice. The other statements can be showed dually. ∎

Note 2.14. Please note that double chain-completeness is only a sufficient condition (under the *Axiom of Choice*) to have a complete multilattice but not a necessary one. Indeed, one can show that the poset depicted in Fig. 2.7 is a complete multilattice (Remark that $\forall i \in \mathbb{N} : c_i \leq a_i$, $c_i \leq c_{i+1}$, $a_i \leq e_0$ and $a_i \leq e_1$) but not chain-complete since the chain $C = \{c_i \mid i \in \mathbb{N}\}$ does not have a join. Indeed, $C^u = \{e_0, e_1\}$ which is an antichain (i.e. C^u has two minimal elements). □

2.3.6 Summary

Table 2.2 summarizes the different poset properties that we have learned so far. Please refer to Table 2.1 of the different notations used here. Fig. 2.8 depicts different posets properties and their relationship. One should note that some links are broken because of the empty set (i.e. the non existence of the top and/or the bottom element). For instance, bounded chain-finite posets are doubly chain-complete.

Last but not least, we invite the reader attention to the notion of the direct product between posets that we will use in this manuscript.

Definition 2.20. The **direct product** of the two posets $(P_1, \leq)$ and $(P_2, \leq)$ is the poset denoted $(P_1, \leq) \times (P_2, \leq)$ and given by:

$$\begin{array}{rcl} (P_1, \leq) \times (P_2, \leq) & := & (P_1 \times P_2, \leq) \\ (x_1, x_2) \leq (y_1, y_2) & \iff & x_1 \leq y_1 \text{ and } x_2 \leq y_2 \end{array}$$

More generally, let I be an arbitrary Index set and let $(P_i, \leq)$ be a poset for each $i \in I$. The **direct product** of posets $(P_i, \leq)$ is the poset denoted by $\times_{i \in I}(P_i, \leq)$ and given by:

$$\begin{array}{rcl} \bigtimes_{i \in I}(P_i, \leq) & := & \left(\bigtimes_{i \in I} P_i, \leq\right) \\ (x_i)_{i \in I} \leq (y_i)_{i \in I} & \iff & (\forall\, i \in I)\ x_i \leq y_i \end{array}$$

Interestingly, the direct product poset preserves all its properties from the starting posets. For instance, if for all $i \in I$, poset $(P_i, \leq)$ are complete lattices then the direct-product poset is also a complete lattice. Moreover:

$$(\forall S \subseteq \bigtimes_{i \in I} P_i)\ \bigwedge S = \left(\bigwedge S_i\right)_{i \in I} \text{ and } \bigvee S = \left(\bigvee S_i\right)_{i \in I} \text{ with } S_i = \left\{x_i \mid (x_j)_{j \in I} \in S\right\}$$

Properties of being bounded, chain-finite, chain-complete or complete multilattices are also transferred to the direct product poset.

Property	**Meaning with** $(P,\leq)$ **poset**
Finite	P is finite
ACC or maximal condition	$(\forall S \in \wp(P))$ S is maximal-handle (i.e. $S \subseteq\downarrow max(C)$)
DCC or minimal condition	$(\forall S \in \wp(P))$ S is minimal-handle (i.e. $S \subseteq\uparrow min(C)$)
Chain-finite	Both properties above (i.e. all chains are finite)
Chain-complete	$(\forall C \in \mathscr{C}(P))$ C has a supremum
Dually Chain-complete	$(\forall C \in \mathscr{C}(P))$ C has an infimum
Doubly Chain-complete	Both properties above
Upper-bounded	Set P has a maximum called the **top** and denoted $\top$
Lower-bounded	Set P has a minimum called the **bottom** and denoted $\bot$
Bounded	Both properties above
Complete Meet-multisemilattice	$(\forall S \in \wp(P))$ S^ℓ is maximal-handle
Complete Join-multisemilattice	$(\forall S \in \wp(P))$ S^u is minimal-handle
Complete Multilattice	Both properties above
Meet-multisemilattice	$(\forall S \in \wp(P) \mid S$ nonempty and finite$)$ S^ℓ is maximal-handle
Join-multisemilattice	$(\forall S \in \wp(P) \mid S$ nonempty and finite$)$ S^u is minimal-handle
Multilattice	Both properties above
Benado's Multilattice	$(\forall p,q \in P)$ $\{p,q\}^\ell$ is maximal-handle and $\{p,q\}^u$ is minimal-handle
Complete lattice	$(\forall S \in \wp(P))$ S^ℓ is maximum-handle and S^u is minimum-handle
Meet-semilattice	$(\forall p,q \in P)$ $\{p,q\}^\ell$ is maximum-handle
Join-semilattice	$(\forall p,q \in P)$ $\{p,q\}^u$ is minimum-handle
Lattice	Both properties above

Table 2.2: Partially ordered set properties

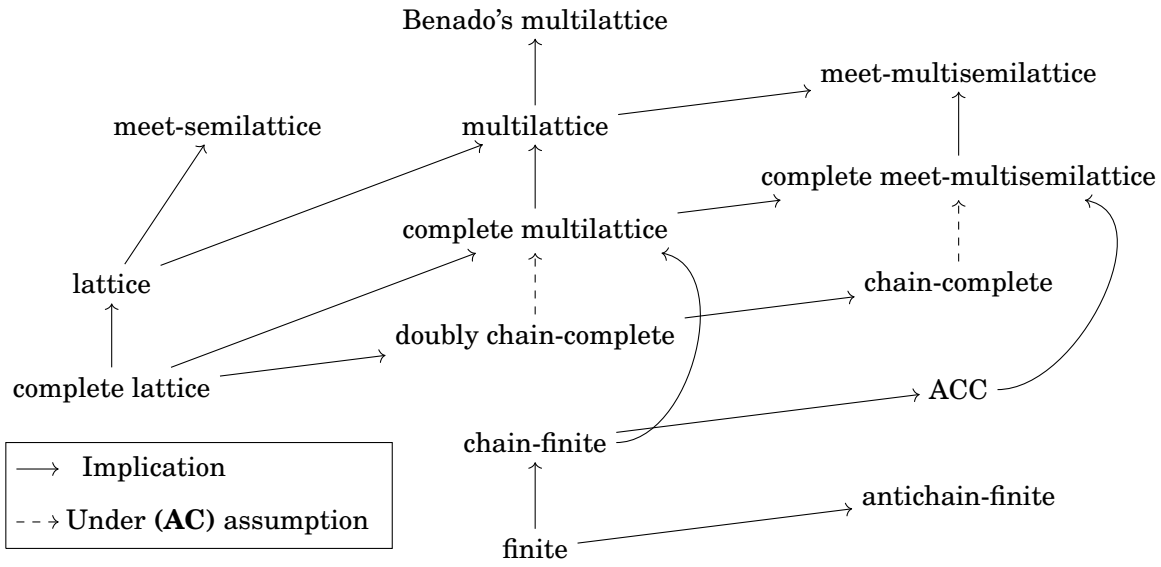


Figure 2.8: Posets properties and their relationships. The dual properties (DCC, join-multisemilattices, etc.) and their relationships can be deduced analgeously.

2.4 Morphisms on Partially Ordered Sets

In this section, we will deal with mappings between partially ordered sets and their properties. We will particularly be interested on the behavior of the mappings when dealing with lattices and complete lattices defined earlier in this chapter.

2.4.1 Basic Definitions

Definition 2.21. Let be the two posets $(P,\leq)$ and $(Q,\leq)$. A mapping $f:P\to Q$ is said to be:

- **Order-preserving** or **monotone**: $(\forall x,y\in P)\ x\leq y\Rightarrow f(x)\leq f(y)$.
- **An Order-embedding**: $(\forall x,y\in P)\ x\leq y\Leftrightarrow f(x)\leq f(y)$
- **An Order-isomorphism**: if f is a surjective order embeddig (i.e. $f[P]=Q$).
- **Order-reversing**: $(\forall x,y\in P)\ x\leq y\Rightarrow f(y)\leq f(x)$
- **An Order anti-embedding**: $(\forall x,y\in P)\ x\leq y\Leftrightarrow f(y)\leq f(x)$

Note 2.15. Please note that:

- Order-preserving and order-reversing mappings present the following problem: even if $x,y\in P$ are incomparable, $f(x)$ and $f(y)$ can become comparable. This is not the case when dealing with order embeddings and order anti-embeddings.
- Order-embedding and order-reversing mappings are injective mappings. Indeed:

$$f(x)=f(y)\Rightarrow f(x)\leq f(y) \text{ and } f(y)\leq f(x)\Rightarrow x\leq y \text{ and } y\leq x\Rightarrow x=y$$

- If there exists an order-embedding $f:P\to Q$ we say that $(Q,\leq)$ is an **embedding** of $(P,\leq)$.
- If there exists an order-isomorphism $f:P\to Q$ we say that $(P,\leq)$ is **order-isomorphic** to $(Q,\leq)$ and we denote $(P,\leq)\cong(Q,\leq)$. Note that sets P and Q are isomorphic too since f is a bijection. One should note also that order-isomorphic posets are technically indistinguishable in the sense that one can study the properties of a poset through some poset that is order-isomorphic to it. □

When dealing with mappings between complete lattices, order-preserving mappings does not preserve necessarily the meet and the join. We have in fact the following proposition:

Proposition 2.3. Let be two complete lattices $(P,\leq)$ and $(Q,\leq)$ and an order-preserving mapping $f:P\to Q$, we have:

$$(\forall S\subseteq P)\ f\left(\bigwedge S\right)\leq\bigwedge f[S] \qquad\qquad (\forall S\subseteq P)\ f\left(\bigvee S\right)\geq\bigvee f[S]$$

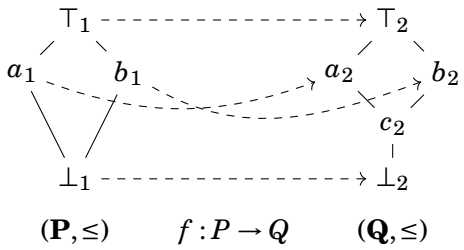


Figure 2.9: An order-embedding that is join-preserving but not meet-preserving

Proof. Let us show $(\forall S \subseteq P)\ f(\bigwedge S) \leq \bigwedge f[S]$. Let $S \subseteq P$, we have $(\forall s \in S)\ \bigwedge S \leq s$. Hence, $(\forall s \in S)\ f(\bigwedge S) \leq f(s)$. It follows that: $(\forall s \in S)\ f(\bigwedge S) \leq f(s)$. Therefore: $f(\bigwedge S) \leq \bigwedge f[S]$. One can follow exactly the same steps to show the other inequality. ∎

Definition 2.22. Let be the two complete lattices $(P,\leq)$ and $(Q,\leq)$. A mapping $f : P \to Q$ is said to be:

- **Meet-preserving** if: $(\forall S \subseteq P)\ f(\bigwedge S) = \bigwedge f[S]$
- **Join-preserving** if: $(\forall S \subseteq P)\ f(\bigvee S) = \bigvee f[S]$

One should note that if a mapping is meet-preserving or join-preserving then it is order-preserving. The following proposition states an important property of order-isomorphismes.

Proposition 2.4. Let be two complete lattices $(P,\leq)$ and $(Q,\leq)$. if $f : P \to Q$ be an order-isomorphism then it is both meet-preserving and join-preserving.

Proof. Let $S \subseteq P$. Since f is order-preserving then $f(\bigwedge S) \leq \bigwedge f[S]$ as shown in Proposition 2.3. Let us show now that $f(\bigwedge S) = \bigwedge f[S]$. Suppose that $f(\bigwedge S) < \bigwedge f[S]$. since f is surjective, we have $(\exists x \in P) f(x) = \bigwedge f[S]$. Hence, $f(\bigwedge S) < f(x)$. Since f is an order-embedding, it follows that $\bigwedge S < x$. On the other hand, we have $(\forall s \in S)\ f(x) < f(s)$. Using again the fact that f is an order-embedding we conclude that $(\forall s \in S)\ x < s$. In other words, $x < \bigwedge S$ leads to a contradiction with the fact that $\bigwedge S < x$ and $<$ is a strict order. One can follow the same steps to show that f is also join-preserving. ∎

Note 2.16. An order-embedding is not necessarily meet-preserving (or join-preserving). Consider for instance the mapping $f : P \to Q$ depicted in Fig. 2.9. It is clear that f is not meet-preserving. Indeed: $\bot_2 = f(\bot_1) = f(a_1 \wedge b_1) \neq f(a_1) \wedge f(b_1) = a_2 \wedge b_2 = c_2$. □

Proposition 2.5. Let $(P,\leq)$ be a complete lattice and $(Q,\leq)$ be an arbitrary poset. if $f : P \to Q$ is an order-embedding then $(f[P],\leq)$ is a complete lattice and for any $X \subseteq P$ we have:

$$\bigwedge_{f[P]} f[X] = f\left(\bigwedge_{P} X\right) \qquad \bigvee_{f[P]} f[X] = f\left(\bigvee_{P} X\right)$$

Clearly, $(P,\leq)$ and $(f[P],\leq)$ are order-isomorphic.

Proof. Let $X \subseteq P$, we need to show that the meet $f[X]$ in $f[P]$ is $f(\bigwedge_P X)$. Formally:

$$(\forall x \in X)\ \left((\forall p \in P)\ f(p) \leq f(x) \iff f(p) \leq f\left(\bigwedge_P X\right)\right)$$

Since f is an order embedding, the above expression is equivalent to:

$$(\forall x \in X)\ \left((\forall p \in P)\ p \leq x \iff p \leq \bigwedge_P X\right)$$

which is true by definition. One can follow the same steps to proof the second part of the proposition. ∎

Note 2.17. Proposition 2.5 shows that order embeddings are still interesting in the sense that they preserve the structures of the starting poset. However, one shoud keep in mind that even if $(Q, \leq)$ and $(f[P], \leq)$ are complete lattices, $(f[P], \leq)$ is not a complete sublattice of $(Q, \leq)$. This property demands to the embedding to be surjective (i.e. isomorphism). □

2.4.2 Endomorphisms on Partially Ordered Sets

An **endomorphism** on a poset $(P, \leq)$ is a morphism $f : P \to P$, i.e. a mapping from a poset to itself. When dealing with such morphisms, other interesting properties arise and play an essential role in lattice theory. Let us start by defining some notions around endomorphisms.

Definition 2.23. Let $(P, \leq)$ be a poset. An endomorphism $f : P \to P$ is said to be:

- **Extensive** $(\forall p \in P)\ p \leq f(p)$
- **Contractive** or **Intensive**: $(\forall p \in P)\ f(p) \leq p$
- **Idempotent**: $(\forall p \in P)\ f(f(p)) = f(p)$.

Moreover, we say that $p \in P$ is a **fixpoint** of f iff $f(p) = p$.

Note 2.18. The set of fixpoints of an idempotent mapping f is given by $f[P] = \{f(p) \mid p \in P\}$. □

We have seen in Example 2.10 that if a poset $(P, \leq)$ is a complete lattice, not any subposet $(S, \leq)$ of $(P, \leq)$ is a complete lattice. Interestingly, we have the following fact.

Proposition 2.6 (Lemma 1 and Proposition 3 in [77]). A subposet $(S, \leq)$ of a complete lattice $(P, \leq)$ is a complete lattice if and only if there exists an order-preserving and idempotent mapping $f : P \to P$ s.t. $f[P] = S$. Moreover, with $\bigvee_P$, $\bigwedge_P$ denoting respectively the meet and join in P and $\bigvee_S$, $\bigwedge_S$ denoting respectively the meet and join in S, we have:

$$(\forall A \subseteq S)\ \bigwedge_S A = f\left(\bigwedge_P A\right) \quad \textbf{and} \quad \bigvee_S A = f\left(\bigvee_P A\right)$$

Note 2.19. In fact, as stated in Corollary 1 in [77], a subposet $(S, \leq)$ of a complete lattice $(P, \leq)$ is a complete lattice iff it is the set of fixpoints of some order-preserving mapping $f : P \to P$. □

Hence order-preserving and idempotent endomorphism on complete lattices induce by their image a complete lattice. However, such a complete lattice does not necessarily preserve the meets or the joins.

Let us consider the following example.

Example 2.15. Consider, poset $(S,\subseteq)$ **(2)** in Fig. 2.4 is a subposet of the complete lattice $(\wp(E),\subseteq)$ with $E=\{a,b,c,d\}$. Clearly, $(S,\leq)$ is a complete lattice but does not preserve meet nor join, i.e. consider $\{a,b\}$ and $\{b,c\}$ which meet in S is $\emptyset$ rather than $\{b\}$ and join is $\{a,b,c,d\}$ rather than $\{a,b,c\}$. Following the proof of Lemma 1 in [77], one can create a mapping f:

$$f:\wp(E)\to\wp(E), A\mapsto \bigvee_{S}\{B\in S \mid B\subseteq A\}$$

which is clearly order-preserving and idempotent with $f[\wp(E)]=S$. For instance, we have $f(\{a,b,c\})=\bigvee_S\{\emptyset,\{a,b\},\{b,c\}\}=\{a,b,c,d\}$ and $f(\{a,b,d\})=\bigvee_S\{\emptyset,\{a,b\}\}=\{a,b\}$. This shows that f is neither extensive nor intensive. □

Adding the property of extensivity or contractivity to an idempotent and order-preserving mappings form well-known operators defined below.

Definition 2.24. Let $(P,\leq)$ be a poset, we have:

- A **Closure** operator $\phi:P\to P$ is an order-preserving, idempotent and extensive mapping on $(P,\leq)$. Intuitively, a closure operator ϕ associates to each element of P the **smallest fixpoint** of ϕ above it. Formally: $(\forall x\in P\ \forall y\in\phi[P])\ x\leq y\Leftrightarrow\phi(x)\leq y$.
- A **Kernel** or **Interior** operator $\psi:P\to P$ is an order-preserving, idempotent and contractive mapping on $(P,\leq)$. Intuitively, a kernel operator ψ associates to each element of P the **largest fixpoint** of ϕ below it. Formally: $(\forall x\in P\ \forall y\in\psi[P])\ y\leq x\Leftrightarrow y\leq\psi(x)$.

We have seen in Proposition 2.6 that the image of a complete lattice by an order-preserving and idempotent operator form a complete lattice. Theorem 2.6 shows that adding the extensivity (resp. intensivity) to the operator makes the image complete lattice a closure (resp. kernel) system.

Theorem 2.6. Let $(P,\leq)$ be a complete lattice, we have:

- For any closure operator $\phi:P\to P$, Poset $(\phi[P],\leq)$ is a closure system.
- For any kernel operator $\psi:P\to P$, Poset $(\psi[P],\leq)$ is a kernel system.

Proof. Let us show that $\phi[P]$ is a closure system that is for $C\subseteq\phi[P]$ we have $\bigwedge C\in\phi[P]$. Let $C\subseteq\phi[P]$, since ϕ is extensive then $\bigwedge C\leq\phi(\bigwedge C)$. On the other hand, we have $(\forall c\in C)\ \bigwedge C\leq c$. Hence, since ϕ is order-preserving and idempotent $(\forall c\in C)\ \phi(\bigwedge C)\leq\phi(c)=c$. Therefore, $\phi(\bigwedge C)\leq\bigwedge C$. We conclude that $\bigwedge C=\phi(\bigwedge C)$ or in other words, $\bigwedge C\in\phi[P]$. One can follow the same steps to show the second part of the theorem. ∎

Dually, one can build closure and kernel operators starting from an arbitrary subset of a complete lattice as shown in Theorem 2.7.

Theorem 2.7. Let $(P,\leq)$ be a complete lattice and let $S \subseteq P$ be an arbitrary subset:

- The mapping ϕ_S given below is a *closure operator* on $(P,\leq)$.

$$\phi_S : P \to P, p \mapsto \phi_S(p) = \bigwedge \{s \in S \mid p \leq s\}$$

  Poset $(\phi_S[P],\leq)$ is the smallest closure system encapsulating S and $\phi_S[P] = \{\bigwedge A \mid A \subseteq S\}$.
- The mapping ψ_S given below is a *kernel operator* on $(P,\leq)$.

$$\psi_S : P \to P, p \mapsto \psi_S(p) = \bigvee \{s \in S \mid s \leq p\}$$

  Poset $(\psi_S[P],\leq)$ is the smallest kernel system encapsulating S and $\psi_S[P] = \{\bigvee A \mid A \subseteq S\}$.

Proof. Let us show that ϕ_S has the three properties of a closure operator:

- ϕ_S *is extensive.* This property is trivial since $p \in \{s \in S \mid p \leq e\}^{\ell}$ and the meet is the greatest element. Thus: $p \leq \phi_S(p)$.
- ϕ_S *is order-preserving.* let $p_1 \leq p_2$ be two elements in P. It is clear that:

$$\{s \in S \mid p_2 \leq s\} \subseteq \{s \in S \mid p_1 \leq s\}$$
$$\{s \in S \mid p_1 \leq s\}^{\ell} \subseteq \{s \in S \mid p_2 \leq s\}^{\ell}$$

  Thus $\phi_S(p_1) \leq \phi_S(p_2)$. Since the l.h.s. of the first is below the maximum of the r.h.s.
- ϕ_E *is idempotent.* In fact we have:

$$\{s \in S \mid \phi_S(p) \leq s\} = \{s \in S \mid p \leq s\}$$

  Inclusion $\subseteq$ holds since $p \leq \phi_S(p)$. The second inclusion $\supseteq$ comes from the definition. Indeed, since $\phi_S(p)$ is a lower bound for $\{s \in S \mid p \leq s\}$, then for any $s \in S$ such that $p \leq s$ we have $\phi_S(p) \leq s$. We conclude the idempotence:

$$\phi_S(\phi_S(p)) = \bigwedge \{s \in S \mid \phi_S(p) \leq s\} = \bigwedge \{s \in S \mid p \leq s\} = \phi_S(p)$$

Let us show now that $\phi_S[P] = \{\bigwedge A \mid A \subseteq S\}$ by double inclusion:

($\subseteq$) Let $c \in \phi_S[P]$, that is $\exists p \in P$ s.t. $c = \phi_S(p)$. Hence $\exists A \subseteq S$ s.t. $c = \bigwedge S$. Thus $c \in \{\bigwedge A \mid A \subseteq S\}$.

($\supseteq$) We have $\forall s \in S$, $\phi_S(s) = s$. Thus $S \subseteq \phi_S[P]$. Since $(\phi_S[P],\leq)$ closure system, we have: $(\forall A \subseteq \phi_S[P]) \bigwedge A \in \phi_E[P]$. Particularly, for $A \subseteq S$, we have $\bigwedge A \in \phi_S[P]$. We conclude that $\{\bigwedge A \mid A \subseteq S\} \subseteq \phi_S[P]$.

One can follow the same steps to prove the second part of the theorem. ∎

An important corollary of Theorem 2.7 is the following.

Corollary 2.1. Let $(P,\leq)$ be a complete lattice and let $(S,\leq)$ be a subposet of $(P,\leq)$, we have:

- If $(S,\leq)$ is a closure system then $\phi_S[P]=S$. Moreover, the join $\bigvee_S$ is given by:

$$(\forall A\subseteq S)\ \bigvee_S A=\phi_S\left(\bigvee_P A\right)$$

  Intuitively, ϕ_S associates to each element in P the smallest element in S above p. One can say that ϕ_S gives the closest approximation of p in S from above.
- if $(S,\leq)$ is a kernel system then $\psi_S[P]=S$. Moreover, the meet $\bigwedge_S$ is given by:

$$(\forall A\subseteq S)\ \bigwedge_S A=\psi_S\left(\bigwedge_P A\right)$$

  Intuitively, ψ_S associates to each element in P the greatest element in S below p. One can say that ψ_S gives the closest approximation of p in S from below.

Note 2.20. Analogously to Proposition 2.6, Corollary 2.1 and Theorem 2.6 together tell that a subposet of a complete lattice is a closure (resp. kernel) system if and only if it is the image of a closure (resp. kernel) operator.

Please note also that with the help of Theorem 2.7, one can define two others closure operator Φ and Ψ on the complete lattice $(\wp(P),\subseteq)$ based respectively on ϕ_S and ψ_S. Intuitively, operator Φ (resp. Ψ) associates to any subset $S\subseteq P$, the smallest closure system $\Phi(S)$ (resp. smallest kernel system $\Psi(S)$) on the complete lattice $(P,\leq)$ enclosing it.

$$\Phi:\wp(P)\to\wp(P), S\mapsto\phi_S[P]=\left\{\bigwedge A\mid A\subseteq S\right\}$$
$$\Psi:\wp(P)\to\wp(P), S\mapsto\psi_S[P]=\left\{\bigvee A\mid A\subseteq S\right\}$$

The fixpoints of Φ (resp. Ψ) are the set of all possible closure (resp. kernel) systems on $(P,\leq)$. □

We have built several complete lattices starting from a morphism that is idempotent and order-preserving. Yet, until now, neither of these complete lattices were complete sublattices. Let us analyze the following example.

Example 2.16. Consider the closure system of $(\wp(a,b,c,d),\subseteq)$ depicted in Fig. 2.4 **(3)** which is clearly not a complete sublattice of $(\wp(a,b,c,d),\subseteq)$. The associated closure operator ϕ_S is:

- **not meet-preserving:** we have: $\phi_S(\{a\}\cap\{c\})=\phi_S(\emptyset)=\emptyset$. On the other hand, $\phi_S(\{a\})\cap\phi_S(\{c\})=\{a,b\}\cap\{b,c\}=\{c\}$. Hence: $\phi_S(\{a\}\cap\{c\})\neq\phi_S(\{a\})\cap\phi_S(\{c\})$.
- **not join-preserving:** we have: $\phi_S(\{a\}\cup\{c\})=\phi_S(\{a,c\})=\phi_S(\{a,b,c,d\})$. On the other hand, $\phi_S(\{a\})\cup\phi_S(\{c\})=\{a,b\}\cup\{b,c\}=\{a,b,c\}$. Hence: $\phi_S(\{a\}\cup\{c\})\neq\phi_S(\{a\})\cup\phi_S(\{c\})$.

Same remarks hold for the kernel operator ψ_S associated to the kernel system depicted in Fig. 2.4 **(4)**. □

Hence, the image of a complete lattice by a closure operator is not necessarily a complete sublattice. Interestingly, to obtain a complete sublattice with a closure operator, one need a join-preserving closure operator as stated in Theorem 2.8.

Theorem 2.8 (Theorem 4.2 in [54]). Let $(P,\leq)$ be a complete lattice. We have:

- If ϕ is a join-preserving closure on $(P,\leq)$ then $(\phi[P],\leq)$ is a complete sublattice of $(P,\leq)$.
- If ψ is a meet-preserving kernel on $(P,\leq)$ then $(\psi[P],\leq)$ is a complete sublattice of $(P,\leq)$.
- If $(S,\leq)$ is complete sublattice of $(P,\leq)$ then ϕ_S and ψ_S are respectively a join-preserving closure operator and meet-preserving kernel operator.

Note 2.21. To sum up, Let $(P,\leq)$ be a complete lattice. Subposet $(S,\leq)$ of $(P,\leq)$ is:

- A complete lattice iff it is the image of P with an order-preserving and idempotent operator.
- A closure system iff it is the image of P with a closure operator.
- A kernel system iff it is the image of P with a kernel operator.
- A complete sublattice iff it is the image of P with a join-preserving closure operator and/or a meet-preserving kernel operator. □

Before leaving this section, let us analyze a usual closure operator on posets.

Proposition 2.7. For any poset $(P,\leq)$, we have:

- Up-closure $\uparrow$ is a join-preserving closure operator on $(\wp(P),\subseteq)$.
- Down-closure $\downarrow$ is a join-preserving closure operator on $(\wp(P),\subseteq)$.

Proof. Let us show that $\uparrow\colon \wp(P)\to\wp(P)$ is a closure operator on $(\wp(P),\subseteq)$. It is clear that $S\subseteq\uparrow S$ (i.e. $\uparrow$ is *extensive*) by definition (see Definition 2.6). For $S\subseteq T$ in $\wp(P)$, we have if $x\in\uparrow S$, then $\exists y\in S\subseteq T$ s.t. $y\leq x$. That is, $x\in\uparrow T$. Thus, $(\forall S,T\in\wp(P))\uparrow S\subseteq T \implies \uparrow S\subseteq\uparrow T$ (i.e. $\uparrow$ is *order-preserving*). Let us show that $\uparrow$ is *idempotent*. It is clear that $\uparrow S\subseteq\uparrow\uparrow S$ since $\uparrow$ is extensive. It remains to show that $\uparrow\uparrow S\subseteq\uparrow S$. Let $x\in\uparrow\uparrow S$, that is $\exists y\in\uparrow S$ such that $y\leq x$. That is $\exists z\in S$ such that $z\leq y\leq x$. We conclude that $x\in\uparrow S$.

Let us show now that $\uparrow$ is also join-preserving. According to Proposition 2.3 and since $\uparrow$ is order-preserving then: $\uparrow(\bigcup_{S\in\mathbb{S}}S)\supseteq\bigcup_{S\in\mathbb{S}}\uparrow S$. Let us show inclusion $\subseteq$. Let $p\in\uparrow(\bigcup_{S\in\mathbb{S}}S)$ then $\exists q\in\bigcup_{S\in\mathbb{S}}S$ s.t. $p\geq q$. By definition of the union, $\exists S\in\mathbb{S}$ s.t. $q\in S$. Therefore, $\exists S\in\mathbb{S}\ \exists q\in S$ $p\geq q$. In other words, $\exists S\in\mathbb{S}$ s.t. $p\in\uparrow S$. Hence, $p\in\bigcup_{S\in\mathbb{S}}\uparrow S$. Therefore, $\uparrow$ is join-preserving. One can follow the same steps to show that $\downarrow\colon\wp(P)\to\wp(P)$ is also a join-preserving closure operator on $(\wp(P),\subseteq)$. ∎

Note 2.22. According to proposition 2.7 and Theorem 2.8 and since $\mathscr{U}(P)$ is the set of fixpoints of $\uparrow$ then $(\mathscr{U}(P),\subseteq)$ is a complete sublattice of $(\wp(P),\subseteq)$ i.e. $(\mathscr{U}(P),\subseteq)$ is closed under both arbitrary intersections and arbitrary unions. Same goes with subposet $(\mathscr{O}(P),\subseteq)$. Moreover, since $(\wp(P),\subseteq)$ are completely distributive (see [80] for the definition), then posets $(\mathscr{U}(P),\subseteq)$ and $(\mathscr{O}(P),\subseteq)$ are completely distributive complete lattices. □

2.4.3 Galois Connections

Galois connections are mappings that were studied thoroughly in the literature (see for example [61, 137]). In this thesis, they provide another way to create closure operators. We define this notion below.

Definition 2.25 (from [137]). Let $(P,\leq)$ and $(Q,\leq)$ be two posets, let $f:P\to Q$ and $g:Q\to P$ be two mappings. The pair (f,g) is said to be a **Galois Connection** iff:

- f and g are order-reversing.
- $f\circ g$ and $g\circ f$ are extenstive.

Equivalently **(cf. proposition 4 in [80])**, the pair (f,g) is said to be a **Galois Connection** iff:

$$(\forall p\in P,\forall q\in Q)\ q\leq f(p)\iff p\leq g(q)$$

Whenever a Galois connection is formed between two posets, two closure operators can be built as shown in the following theorem.

Theorem 2.9 (Proposition 5 and Proposition 8 in [80]). Following the notation of Definition 2.25 and for (f,g) a Galois connection, mappings $f\circ g$ and $g\circ f$ are respectively closure operators on $(P,\leq)$ and $(Q,\subseteq)$. Moreover, we have $f=f\circ g\circ f$ and $g=g\circ f\circ g$.

Note 2.23. Please note that the set of fixpoints of $f\circ g$ is given by $g[Q]$ and the set of fixpoints of $g\circ f$ is given by $f[P]$. Hence, if $(P,\leq)$ and $(Q,\leq)$ are complete lattices, $(f[P],\leq)$ and $(g[Q],\leq)$ are respectively closure systems in $(Q,\leq)$ and $(P,\leq)$ (see Theorem 2.6). □

We have seen in Proposition 2.7 that up-closure $\uparrow$ and down-closure $\downarrow$ are join-preserving closure operators. The proposition below states also an important fact about the pair of operators $\left(^{\ell},^{u}\right)$ (see Definition 2.7).

Proposition 2.8. For any poset $(P,\leq)$, the pair $\left(^{\ell},^{u}\right)$ forms a Galois connection on $(\wp(P),\subseteq)$.

Proof. We need to show the different properties of a Galois connection. Let $A,B\in\wp(P)$ s.t. $A\subseteq B$. Let $x\in B^{\ell}$, hence $(\forall b\in B)\ x\leq b$. Since $A\subseteq B$, then $(\forall b\in A)\ x\leq b$. Therefor, $x\in A^{\ell}$. In other words, $^{\ell}$ is order-reversing. Same steps can be followed to show that ^{u} is also order-reversing. Let us show now that for any $A\in\wp(P)$, we have $A\subseteq(A^{\ell})^{u}$. This property is trivial for the bottom element $\emptyset$. Suppose now that $A\neq\emptyset$ and let $a\in A$. By definition, we have $(\forall b\in A^{\ell})\ b\leq a$, in other words, $a\in(A^{\ell})^{u}$. Same proof can be done to show that $^{u}\circ^{\ell}$ is also extensive. This concludes the proof. ∎

Since $\left(^{\ell},^{u}\right)$ is a Galois connection on $(\wp(P),\subseteq)$. Fixpoints of the related closure operators $^{u}\circ^{\ell}$ and $^{\ell}\circ^{u}$ form closure systems. Such closure systems are very important ones in Lattice theory and will be discussed in the following section.

2.4.4 Poset Completions

Let $(P,\leq)$ and $(Q,\leq)$ be two posets. We say that $(Q,\leq)$ is an **embedding** of $(P,\leq)$ iff there exists an order-embedding $f : P \to Q$. Moreover, since order-embeddings are injective mappings, we say that $(P,\leq)$ is **smaller**[5] than $(Q,\leq)$.

Definition 2.26 (from [149]). Let $(P,\leq)$ be a poset and $(L,\leq)$ be a complete lattice, if $(L,\leq)$ is an embedding of $(P,\leq)$ then $(L,\leq)$ is said to be a **completion** of $(P,\leq)$.

The most basic completion of a poset $(P,\leq)$ is the complete lattice $(\wp(P),\subseteq)$ where the order-embedding is $\varphi : P \to \wp(P), P \mapsto \downarrow p$. We will present now other usual completions which are the **Dedekind MacNeille Completion** and the **Alexandrov completion**. We will investigate after in more details the **Antichain completion** and its relationship with multilattices.

2.4.4.1 Dedekind-MacNeille completion

Definition 2.27. The **Dedekind-MacNeille Completion** of $(P,\leq)$ is the poset $(DM(P),\subseteq)$ where: $DM(P) := \{S^\ell \mid S \subseteq P\}$. The order-embedding φ is given by: $\varphi : P \to DM(P), p \mapsto \downarrow p$.

Interestingly, the Dedekind-MacNeille completion is the **smallest completion** of a poset (cf. Theorem 5.3.8 in [149]). By smallest, we mean that for any other completion $(L,\leq)$ of a poset, one can embeds the Dedekind-MacNeille completion $(DM(P),\leq)$ into the completion $(L,\leq)$. Therefore, if $(P,\leq)$ is a complete lattice then it is order-isomorphic to $(DM(P),\leq)$. Which is a clear statement since in a complete lattice $(P,\leq)$, S^ℓ are principal ideals for any $S \subseteq P$, i.e. $DM(P) = \phi[P]$.

One should notice also that since $\left(^\ell, ^u\right)$ form a Galois connection on $(\wp(P),\subseteq)$ (see Proposition 2.8), $DM(P)$ represents simply the set of fixpoints of the mapping $(\cdot)^{u\ell}$.

2.4.4.2 Alexandrov completion

Definition 2.28. The **Alexandrov Completion** of $(P,\leq)$ is the poset $(\mathcal{O}(P),\subseteq)$ where $\mathcal{O}(P)$ is the set of lower-ideals of $(P,\leq)$. The order-embedding φ is given by: $\varphi : P \to \mathcal{O}(P), p \mapsto \downarrow p$.

Please notice that since $(\mathcal{O}(P),\subseteq)$ is a complete sublattice of $(\wp(P),\subseteq)$ then it is completely distributive. Hence, the Alexandrov completion provides a completely distributive completion of a given poset rather than the Dedekind-MacNeille one which is not necessarily distributive. But, as said beforehand, Dedekind-MacNeille completion is *smaller* than the Alexandrov one.

[5] The relationship *smaller* is a preorder between sets. Moreover, if P is smaller than Q and Q is smaller than P then P and Q are isomorphic thanks to Schröder–Bernstein theorem.

2.4.4.3 Antichain completion

Definition 2.29. The **antichain embedding** of $(P,\leq)$ is the poset $(\mathscr{A}(P),\leqq)$ s.t.:

- $\mathscr{A}(P)$ is the set of all antichains of $(P,\leq)$.
- The order $\leqq$ is given by $(\forall A,B\in\mathscr{A}(P))\ A\leqq B\Leftrightarrow{\downarrow}A\subseteq{\downarrow}B$.[6]
- The order embedding φ from $(P,\leq)$ to $(\mathscr{A}(P),\leqq)$ is given by

$$\varphi:P\rightarrow\mathscr{A}(P),a\mapsto\{a\}$$

Conversely to the aforementioned completions, the antichain embedding is not necessarily a completion. Works [27] and [51] had discussed the properties of such an embedding. In fact, when P has the **ACC**, $(\mathscr{A}(P),\leq)$ is a *distributive lattice*, where the meet and the join are given by $S_1\wedge S_2=max({\downarrow}S_1\cap{\downarrow}S_2)$ and $S_1\vee S_2=max(S_1\cup S_2)$, respectively. Moreover, $(\mathscr{A}(P),\leqq)$ is always a $\vee$-semilattice whatever the nature of the poset $(P,\leq)$, but not necessarily a lattice. Paper [27] formulated a sufficient and necessary condition in order to have $(\mathscr{A}(P),\leqq)$ be a lattice: $\forall A,B\in\mathscr{A}(P)\ \exists C\in\mathscr{A}(P)\ \ {\downarrow}A\cap{\downarrow}B={\downarrow}C$.

We take the opportunity here to underline an important link between the antichain embedding and multilattices. Before expliciting this link, let us take a close look to the following Lemma.

Lemma 2.3. Let $(P,\leq)$ be a poset and let $\mathscr{A}(P)$ be the set of its antichain. We have $\forall S\subseteq P$:

$$(\exists C\in\mathscr{A}(P))\ S={\downarrow}C\Rightarrow C=max(S)$$

Proof. The case of $S=\emptyset$ is trivial since ${\downarrow}\emptyset=\emptyset$ and $max(\emptyset)=\emptyset$. Let be a nonempty set $S\subseteq P$ s.t. $(\exists C\in\mathscr{A}(P))S={\downarrow}C$. Let us show that $C=max(S)$:

- $C\subseteq max(S)$: let $c\in C\subseteq S$, suppose that $c\notin max(S)$ that is $\exists x\in S$ s.t. $c<x$. Since $S={\downarrow}C$ then $\exists c_2\in C$ s.t. $x\leq c_2$. Thus $\exists c_2\in C$ such that $c>c_2$ which is a contradiction with the fact that C is an antichain.
- $C\supseteq max(S)$: Suppose $\exists a\in max(S)$ s.t. $a\notin C$. We have $a\in S={\downarrow}C$, that is: $\exists c\in C$ s.t. $a<c$ (since $a\notin C$). However, since $S={\downarrow}C$ then $C\subseteq S$. Thus, $\exists c\in S$ s.t. $a<c$ which is in contradiction with the fact that $a\in max(S)$.

This concluds the proof. ∎

Proposition 2.9. Let $(P,\leq)$ be a poset and let $(\mathscr{A}(P),\leqq)$ be its antichain embedding:

- If $(\mathscr{A}(P),\leqq)$ is a lattice then $(P,\leq)$ is a meet-multisemilattice. Moreover, if $(\mathscr{A}(P),\leqq)$ has a top element then $P={\downarrow}max(P)$.
- if $(\mathscr{A}(P),\leqq)$ is a complete lattice, i.e. a **completion** of $(P,\leq)$, then $(P,\leq)$ is a complete meet-multisemilattice.

[6] Note that $\leqq$ does not induce an order in $\wp(P)$, but just a *pre-order*, since the *anti-symmetry* does not hold (see [51]). Indeed, consider poset $(\{a,b\},\leq)$ where $a\leq b$. Since ${\downarrow}\{a,b\}={\downarrow}\{b\}=\{a,b\}$, we have $\{a,b\}\leqq\{b\}$ and $\{b\}\leqq\{a,b\}$. Yet, $\{a,b\}\neq\{b\}$. Therefor, $\leqq$ is not antisymmetric on $\wp(\{a,b\})$ but it is still reflexive and transitive.

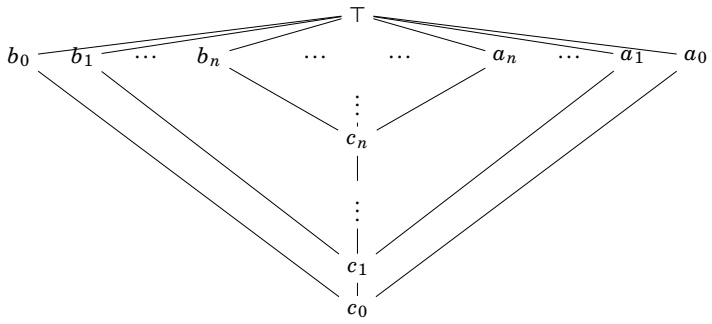


Figure 2.10: The Antichain embedding of this complete lattice is not even a meet-semilattice

Proof. Let us start by showing the first property that is if $(\mathscr{A}(P), \leqq)$ is a lattice then $(P, \leq)$ is meet-multisemilattice. We have $(\mathscr{A}(P), \leqq)$ is a lattice. Then we have:

$$\forall \mathbb{S} \subseteq \mathscr{A}(P) \text{ finite and nonempty} \quad \exists C \in \mathscr{A}(P) \quad \bigcap_{A \in \mathbb{S}} \downarrow A = \downarrow C$$

Since C is an antichain, we have $C = max(\bigcap_{A \in \mathbb{S}} \downarrow A)$ (Lemma 2.3), that is:

$$\forall \mathbb{S} \subseteq \mathscr{A}(P) \text{ finite and non empty} \qquad \bigcap_{A \in \mathbb{S}} \downarrow A = \downarrow max\left(\bigcap_{A \in \mathbb{S}} \downarrow A\right) \tag{2.2}$$

Let be $S \subseteq P$ be a non empty finite subset. We need to show that $S^\ell = \downarrow max(S^\ell)$. We have $S^\ell = \bigcap_{s \in S} \downarrow \{s\}$. Since, $(\mathscr{A}(P), \leqq)$ is a lattice we have according to equation (2.2):

$$S^\ell = \bigcap_{s \in S} \downarrow \{s\} = \downarrow max(S^\ell)$$

If $\mathscr{A}(P)$ is also upper-bounded, that is $\exists C \in \mathscr{A}(P)\ P = \downarrow C$. Hence, we have $P = \downarrow max(P)$ since C is an antichain (Lemma 2.3). In other words, $\emptyset$ has all its multi-infima.

One can follow the same steps to show the second statement of the theorem where $\mathbb{S}$ in Equation 2.2 become now an arbitrary set. ∎

Note 2.24. One should note that the converse of Proposition 2.9 is not true. In fact, one can create complete lattices for which the antichain completion is not even a lattice. Fig. 2.10 depicts such a complete lattice. Indeed, for antichains $A = \{a_i \mid i \in \mathbb{N}\}$ and $B = \{b_i \mid i \in \mathbb{N}\}$, we have $\downarrow A \bigcap \downarrow B = \{c_i \mid i \in \mathbb{N}\}$, i.e. an infinitely ascending chain. Hence, $max(\downarrow A \bigcap \downarrow B) = \emptyset$. □

CHAPTER 3

PATTERNS AS ORDERED SETS

Pattern mining is a general data mining task which aim to discover useful and actionable patterns in databases. Different subtasks can be identified in this field: Frequent Pattern Mining [94], (Class) Association Rule Mining [4, 121], Discriminative Subgroup Discovery [106, 134, 163], Exceptional Model Mining [58, 117], Redescription Mining [75, 147] and High Utility Pattern Mining [70] among others. These different tasks share in their core definition common settings as for instance the pattern language they need to explore: Itemsets [4], Intervals [104], Convex polygons [20], Neighborhood patterns [86], (Complex) Sequential patterns [5, 39, 45, 143], Graph patterns [110, 164], Trajectory patterns [87], Periodic patterns [74, 138], Subgraph patterns in attributed graphs [23, 105], etc. More generally, one should answer the following questions in order to instantiate a pattern mining task:

(1) What is the initial representation, the schema, of the provided database? That is: What are the objects (rows) and what are the descriptive attributes (columns)?
(2) What is the considered pattern language used to describe objects in the database w.r.t. the pattern mining task and the provided information?
(3) How to check whether a pattern holds for some object in the database?
(4) How to evaluate the "interestingness" of the findings (i.e. a quality measure, constraints that patterns or the pattern set need to have, etc.)?

Several formal tools are proposed in the literature to formalize pattern languages or more generally the settings of a pattern mining task. For instance, Formal Concept Analysis (FCA) [80, 162], Symbolic Data Analysis (SDA) [3, 36], Inductive Database [100, 127], Logical Concept Analysis (LCA) [66], Pattern Structures [78, 112] and Relational Concept Analysis (RCA) [93] are frameworks designed to formalize a elements of a pattern mining task. These frameworks rely in general on the fact that they consider pattern languages as *partially ordered sets (posets)*.

Based on the preceding, this chapter aims to provide a better understanding of pattern languages from an order-theoritic point of view. We will not address a particular pattern mining task in this chapter. The study of a particular one, namely *discriminative subgroup discovery*, is left to the third and last part of the dissertation. This chapter is organized as follow:

- **Section 3.1** presents a small introduction on pattern languages.
- **Section 3.2** presents a first broad framework, namely **Formal Concept Analysis (FCA)**, to models pattern languages. Such a framework has been proposed in [162] to provide a conceptual vision on lattices and showed a great utility to formalize itemset pattern language [4] or other more complex ones thanks to conceptual scaling [79].
- **Section 3.3** presents a large framework, namely **pattern setups** defined in [123] which model pattern languages as partially ordered sets. We proposed in [18] a more detailed investigations on this framework that we will develop in this section.
- **Section 3.4** presents the framework of **pattern structures** proposed in [78]. In a nutshell, this framework requires the existence of the "closed" patterns. While this framework models technically the same kind of languages as Formal Concept Analysis does, its usage is more natural to model a plethora of pattern languages that are lattice-bases such as interval patterns [104], convex polygon patterns [20], sequence sets [39], graph sets [110] and partition pattern language [49] among others.
- **Section 3.5** explores particularly the notion of closed descriptions when generalized to other kind of pattern languages than lattice-based ones, i.e. usage of maximal common patterns. We show that pattern setups have several issues with them when no additional condition is required on the partially ordered set of patterns.
- **Section 3.6** presents the new framework of **pattern multistructure** proposed in [18] under the name of *pattern hyper-structure* and named so in [19]. This framework lies between *pattern setups* which rely on arbitrary posets and *pattern structures* which rely on lattices. Pattern multistructures rely in fact on multilattices (see Section 2.3.5) and guarantee that the "closed" patterns induces all subsets separable by the pattern language. This latter property does not necessarily hold in an arbitrary pattern setup. Sequential [6, 166] and graph patterns [165] induce pattern multistructures but not pattern structures.
- **Section 3.7** presents how other pattern setups can be built starting from base ones. We will mainly investigate the following tasks: (1) the task of transforming pattern setups to pattern structures, namely **completions** [39, 78], (2) the task of simplifying pattern languages thanks to **projections** [78] and (3) the task of combining many pattern languages by conjunction using the **direct product**. We will also show some important results linked to these three tasks when pattern multistructures are considered.
- **Section 3.8** sums up the notions presented in this chapter and provide concluding remarks.

The writing of this chapter relies principally on paper [19] currently under review. A preliminary version of this chapter had appeared in paper [18].

3.1 Elements on Pattern Languages

Knowledge Discovery in general or pattern mining in particular starts from a dataset which formal definition is presented below.

Definition 3.1. A **dataset** is a pair $(\mathcal{G}, \mathcal{M})$ where $\mathcal{G}$ is a set of **objects** (i.e rows) and $\mathcal{M}$ is a set of **attributes** (i.e columns)[1]. An **attribute** $m \in \mathcal{M}$ is a mapping $m : \mathcal{G} \to \mathcal{R}_m, g \mapsto m(g)$ that takes each object g to its value $m(g)$. A *dataset* is said to be **finite** iff sets $\mathcal{G}$ and $\mathcal{M}$ are finite.

Note 3.1. Different nature of attributes are considered in a dataset. For instance, an attribute $m \in \mathcal{M}$ is said to be:

- **Ordinal** if its range $\mathcal{R}_m$ is assumed to be totally ordered by some order $\leq$.
- **Numerical** if it is ordinal with $\mathcal{R}_m = \mathbb{R}$.
- **Nominal** or **categorical** if no order is assumed between the different values of $\mathcal{R}_m$.
- **Boolean** if $\mathcal{R}_m = \{true, false\}$. □

Example 3.1. Table 3.1 depicts a finite dataset where $\mathcal{G} = \{g_1, g_2, g_3, g_4, g_5\}$ represents transactions and $\mathcal{M} = \{Cheese, Bread, Apple, Wine\}$ representing products. There is a cross on a cell (g, m) if $m(g) = true$, i.e. the transaction $g \in \mathcal{G}$ has the product $m \in \mathcal{M}$. Otherwise, $m(g) = false$. This dataset is typically the type of dataset studied in the seminal paper [4] on pattern mining. □

Let us now give a first definition on what a pattern is.

Definition 3.2. Let $(\mathcal{G}, \mathcal{M})$ be a dataset. A **pattern** or a **description** $p(x)$ is a first-order predicate with one non-quantified (i.e. free) variable representing the object, i.e. the domain of x is $\mathcal{G}$. The predicate is expressed using the attributes in $\mathcal{M}$.

An object $g \in \mathcal{G}$ **matches** a pattern $p(x)$ if $p(x)$ is valued to true when x is assigned to g. The **extent** of a pattern p is then the set of objects in $\mathcal{G}$ matching the pattern p, i.e.

$$ext(p) = \{g \in \mathcal{G} \mid p(g) = true\}$$

Example 3.2. Consider again the dataset depicted in Table 3.1. The following pattern

$$p(x) := \big(Cheese(x) \wedge Wine(x)\big)$$

has for extent $ext(p) = \{g_3, g_4\}$. That is only g_3 and g_4 matches pattern p. □

Example 3.2 presents an example of a pattern. Clearly, not all possible predicates are considered in a pattern mining tasks. It is in general reduced to a well-identified subset of patterns said to be the **pattern language** of the task. For instance patterns like the ones presented in Example 3.2 and Example 3.3 are conjunction of *"attribute is equal to a value"* constraints. Such a pattern language is generally identified under the name of **itemset pattern language** [4].

[1] $\mathcal{G}$ and $\mathcal{M}$ comes respectively from the German words *Gegenstände* and *Merkmale* for *objects* and *attributes*.

Transactions	Cheese	Bread	Apple	Wine
g_1	×		×	
g_2		×	×	
g_3	×		×	×
g_4	×			×
g_5			×	

Table 3.1: A sample of some supermarket database. All attributes in this dataset are Boolean.

A plethora of other pattern languages has been identified in the literature. One can cite for instance: Interval patterns [104], Convex polygons [20], Neighborhood patterns [86], (Complex) Sequential patterns [5, 39, 45, 143], Trajectory patterns [87], Periodic patterns [74, 138], Subgraph patterns [23, 105] among others. Some examples has been presented earlier in the general introduction (see Example 1.2 and Fig. 1.2). Clearly, choosing a pattern language depends on the provided dataset and the targeted task.

Pattern language can be seen simply as a set of predefined patterns. A stronger way to organize it is to see it as a an **ordered set**. This can be done using the implications existing between the different patterns and considering equivalent patterns as the same one. Consider for instance the following example:

Example 3.3. Let be the following predicate:

$$q(x) := Wine(x)$$

It is clear that we have $p(x) \rightarrow q(x)$ where $p(x)$ is the pattern presented in Example 3.2. We will say that $p(x)$ is **more restrictive** than $q(x)$. Notice that every object matching p matches q but not the converse, i.e. $ext(p) \subseteq ext(q)$. □

Example 3.3 shows that patterns can be **ordered** via the implications existing between them. Following this observation, we will consider **pattern languages** as **ordered sets**.

Our aim in this chapter is to provide an *order-theoritic point of view on pattern languages*. This point of view helps to understand what are the commonalities and the differences between pattern languages. We will start by presenting Formal Concept Analysis (FCA) framework [80, 162] then explore more generic frameworks: pattern setups [18, 123], pattern structures [78, 112] and pattern multistructures [18]. Often, we will invoke without better explanations definitions and notations introduced in Chapter 2. Some of these notations are resumed in Table 2.1.

3.2 Formal Concept Analysis

Formal Concept Analysis (FCA) was introduced by *Wille* in [162] as a mathematical field that provided a conceptual vision on complete lattices. It proved then its usefulness as a tool to analyze Datasets. We present here a small overview on *FCA*. More details can be found in [80].

3.2.1 Basic Definitions

Formal Concept Analysis (FCA) starts by a *formal context* defined below.

Definition 3.3. A **(formal) context** is a triple $\mathbb{K} = (\mathcal{G}, \mathcal{M}, I)$ where $\mathcal{G}$ is a set of objects, $\mathcal{M}$ is a set of attributes and I is a binary relation on $\mathcal{G} \times \mathcal{M}$ called the *Incidence relation*. For $(g,m) \in \mathcal{G} \times \mathcal{M}$, $g\,I\,m$ holds iff g **has attribute** m.

From the dataset perspective (see Definition 3.1), a formal context can be seen as a dataset where all the attributes are Booleans.

Example 3.4. Fig. 3.1 depicts a formal context $(\mathcal{G}, \mathcal{M}, I)$. The incidence relation I is represented by the crosses in the table. For instance, we have $g_4\ I\ b$, i.e. g_4 has attribute b. □

Two base derivation operators stem from a formal context and are defined below.

Definition 3.4. The **extent operator**, denoted ext, associates to each itemset $B \subseteq \mathcal{M}$ the set of objects $g \in \mathcal{G}$ having all items in B. Formally, it is given by:

$$ext : \wp(\mathcal{M}) \rightarrow \wp(\mathcal{G}), B \mapsto \{g \in \mathcal{G} \mid (\forall m \in B)\, g\, I\, m\}$$

The set of all possible extents of a context $\mathbb{K}$ is denoted $\mathbb{K}_{ext}$ and is given by $\mathbb{K}_{ext} = ext[\wp(\mathcal{M})]$.

Definition 3.5. the **intent operator**, denoted int, associates to each subset of objects $A \subseteq \mathcal{G}$ the set of items $m \in \mathcal{M}$ common to the objects in A. Formally, it is given by:

$$int : \wp(\mathcal{G}) \rightarrow \wp(\mathcal{M}), A \mapsto \{m \in \mathcal{M} \mid (\forall g \in A)\, g\, I\, m\}$$

The set of all possible intents of a context $\mathbb{K}$ is denoted $\mathbb{K}_{int}$ and is given by $\mathbb{K}_{int} = int[\wp(\mathcal{G})]$.

Proposition 3.1. The pair (ext, int) forms a *Galois connection* between $(\wp(\mathcal{M}), \subseteq)$ and $(\wp(\mathcal{G}), \subseteq)$

Proof. We show the four properties (see Definition 2.25) below:

- *ext is order-reversing:* let $B_1 \subseteq B_2 \in \wp(\mathcal{M})$. Let us show that $ext(B_2) \subseteq ext(B_1)$. Let $g \in ext(B_2)$ then $(\forall m \in B_2)\, g\, I m$. Since $B_1 \subseteq B_2$ then $(\forall m \in B_1)\, g\, I m$. Hence, $g \in ext(B_1)$.
- *$int \circ ext$ is extensive:* Let $B \in \wp(\mathcal{M})$, we need to show that $B \subseteq int(ext(B))$. Let $m \in B$, we have: $(\forall g \in ext(B))\ g\, I m$. Therefore, $m \in int(ext(B))$.

One can follow the same steps to show that int is order-reversing and $ext \circ int$ is extensive.

$\mathcal{G}$	a	b	c
g_1	×	×	×
g_2	×		
g_3	×		
g_4		×	×

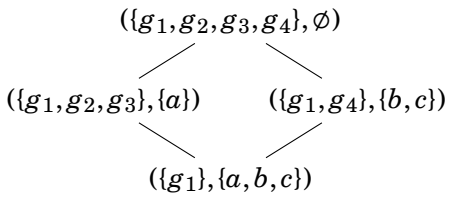


Figure 3.1: Formal Context $(\mathcal{G}, \mathcal{M}, I)$ and its concept lattice $\underline{\mathfrak{B}}(\mathcal{G}, \mathcal{M}, I)$

Since (ext, int) is a Galois connection. Mappings $ext \circ int$ and $int \circ ext$ are closure operators (see Theorem 2.9). Fixpoints of the closure operator $ext \circ int$, given by $\mathbb{K}_{ext} = ext[\wp(\mathcal{M})]$, are called **extents**. Dually, fixpoints of the closure operator $int \circ ext$, given by $\mathbb{K}_{int} = int[\wp(\mathcal{G})]$, are called **intents** or **closed itemsets** in pattern mining literature [139]. Moreover, using the fact that the image of closure operators creates closure systems (i.e. complete lattices preserving the meet, the intersection), posets $(\mathbb{K}_{ext}, \subseteq)$ and $(\mathbb{K}_{int}, \subseteq)$ are complete lattices where the meet is set intersection. An isomorphic lattice to the complete lattice $(\mathbb{K}_{ext}, \subseteq)$ is called the **concept lattice**.

Definition 3.6. The **concept lattice** associated to the formal context $(\mathcal{G}, \mathcal{M}, I)$ is the *complete lattice* denoted by $\underline{\mathfrak{B}}(\mathcal{G}, \mathcal{M}, I) = (\mathfrak{B}(\mathcal{G}, \mathcal{M}, I), \leq)$. Elements of $\mathfrak{B}(\mathcal{G}, \mathcal{M}, I)$ are called **(formal) concepts** and are given by:

$$(A, B) \in \wp(\mathcal{G}) \times \wp(\mathcal{M}) \quad s.t. \quad A = ext(B) \text{ and } B = int(A)$$

The concepts are ordered by $\leq$ as follows:

$$(A_1, B_1) \leq (A_2, B_2) \Leftrightarrow A_1 \subseteq A_2 \Leftrightarrow B_2 \subseteq B_1$$

Example 3.5. Fig. 3.1 (right) depicts the Hasse Diagram of the concept lattice $\mathfrak{B}(\mathcal{G}, \mathcal{M}, I)$. For instance $(\{g_1, g_4\}, \{b, c\})$ is a formal concept, i.e. objects in $\{g_1, g_4\}$ have in common attributes in $\{b, c\}$ and attributes b and c hold both for g_1 and g_4. □

Note 3.2. One should notice that by definition we have:

$$(\forall B_1, B_2 \subseteq \mathcal{M})\ ext(B_1 \cup B_2) = ext(B_1) \cap ext(B_2)$$
$$(\forall A_1, A_2 \subseteq \mathcal{M})\ int(A_1 \cup A_2) = int(A_1) \cap int(A_2)$$

□

For ease of notation, for any $g \in \mathcal{G}$, we call **object intent** of g, the set denoted by $int(g)$ and given by $int(\{g\})$, i.e. the set of attributes holding for g. Analogously, for any $m \in \mathcal{M}$, we call **attribute extent** of m, the set denoted by $ext(m)$ and given by $ext(\{m\})$, i.e. the set of objects for which attribute m hold. We have:

$$(\forall B \subseteq \mathcal{M})\, ext(B) = \bigcap_{m \in B} ext(m) \quad \text{and} \quad (\forall A \subseteq \mathcal{G})\, int(A) = \bigcap_{g \in A} int(g)$$

Note 3.3. Please note that we have $ext(\emptyset) = \mathcal{G}$ and $int(\emptyset) = \mathcal{M}$. □

Another important notion in FCA tightly linked to formal concepts and closed itemsets are implications.

Definition 3.7. Let be two itemsets $B_1, B_2 \in \mathcal{M}$, we say that B_1 **implies** B_2 and we denote $B_1 \rightarrow B_2$ iff $ext(B_1) \subseteq ext(B_2)$. In other words, if an object has all items in the set of attributes B_1 then it has all items in itemset B_2. If we have $B_1 \rightarrow B_2$ and $B_2 \rightarrow B_1$ we say that B_1 and B_2 are **equivalent** and we have $ext(B_1) = ext(B_2)$.

Implications and equivalence between sets of object can be defined analogously.

Example 3.6. In the formal context considered in Fig. 3.1, itemset $\{a,b\}$ implies $\{c\}$ since $ext(\{a,b\}) = \{g_1\} \subseteq \{g_1, g_4\} \subseteq ext(\{c\})$. We write $\{a,b\} \rightarrow \{c\}$. Notice also that item(set)s $\{b\}$ and $\{c\}$ are equivalent since they have the same extent. □

From this notions of equivalence between itemsets, one can notice that each extent $A \in \mathbb{K}_{ext}$ has many itemsets generating it, i.e. itemsets $B \in \wp(\mathcal{M})$ s.t. $B = ext(A)$.

Definition 3.8. Let $A \in \mathbb{K}_{ext}$. An itemset $B \in \wp(\mathcal{M})$ is said to be a **generator** of A iff $ext(B) = A$. It is said to be a **minimal generator** iff:

$$(\forall B' \subsetneq B)\ ext(B') \subsetneq ext(B)$$

Example 3.7. Consider the extent $A = \{g_1, g_4\}$ in the formal context considered in Fig. 3.1. The generators of A are given by $\{b\}$, $\{c\}$ and $\{b,c\}$. It is clear that A has two minimal generators given by $\{b\}$ and $\{c\}$. □

Note 3.4. There is always a **maximum generator** for $A \in \mathbb{K}_{ext}$ in a formal context. It is given by the closed itemset $int(A)$. Indeed, $ext(int(A)) = A$ since A is a fixpoint for the closure operator $ext \circ int$. And for any $B \in \wp(\mathcal{M})$ s.t. $ext(B) = A$ (i.e. generators of A), it is clear $B \subseteq int \circ ext(B) = int(A)$ by extensivity of the closure operator $int \circ ext$.

As shows in Example 3.7, there may by many **minimal generators** for the same extent A. These minimal itemsets can be seen as minimal generators for the closure operator $int \circ ext$ following Definition 4.5 in Chapter 4. Interestingly, while the set of all *maximum generators* forms a closed-under-intersection set system $\mathbb{K}_{int}$, the set of all *minimal generators* forms an independent set system as noticed by [151] (see Proposition 4.9). □

3.2.2 Context Clarification

We have seen in Example 3.6 that attributes b and c have the same extent. Hence, attributes b and c could be considered indistinguishably as a single attribute $d := b \wedge c$. Same remarks hold for some row in the context depicted in Fig. 3.1. Indeed, we can say that objects g_2 and g_3 are equivalent since they have the same intent.

$\mathcal{G}$	$\{a\}$	$\{b,c\}$
g_1	×	×
g_2	×	
g_3	×	
g_4		×

$\mathcal{G}$	a	b	c
$\{g_1\}$	×	×	×
$\{g_2,g_3\}$	×		
$\{g_4\}$		×	×

$\mathcal{G}$	$\{a\}$	$\{b,c\}$
$\{g_1\}$	×	×
$\{g_2,g_3\}$	×	
$\{g_4\}$		×

Figure 3.2: From left to right. a column clarification, a row clarification and a clarification of the context $(\mathcal{G},\mathcal{M},I)$ depicted in Fig. 3.1

Definition 3.9. A context $(\mathcal{G},\mathcal{M},I)$ is said to be:

- **column clarified** iff $\forall m_1,m_2\in\mathcal{M}$, if $ext(m_1)=ext(m_2)$ then $m_1=m_2$.
- **row clarified** iff $\forall g_1,g_2\in\mathcal{G}$, if $int(g_1)=int(g_2)$ then $g_1=g_2$.
- **clarified** iff it is both column clarified and row clarified.

Example 3.8. The formal context depicted in Fig. 3.1 is neither column clarified nor row clarified since $ext(b)=ext(c)$ and $int(g_2)=int(g_3)$. □

The action of transforming a non-clarified context to a clarified one is called **clarification** and is defined formally below.

Definition 3.10. Let $(\mathcal{G},\mathcal{M},I)$ be a context and let $\leftrightarrow_{\mathcal{M}}$ be the equivalence relation on $\mathcal{M}$ defined as follow:

$$(\forall m_1,m_2\in\mathcal{M})\ m_1\leftrightarrow_{\mathcal{M}} m_2\ \ \textbf{iff}\ \ ext(m_1)=ext(m_2)$$

The **column clarification** of $(\mathcal{G},\mathcal{M},I)$ is the column-clarified formal context $(\mathcal{G},\mathcal{M}',I')$ where $\mathcal{M}'$ is given by the quotient set $\mathcal{M}/\leftrightarrow_{\mathcal{M}}$ and I' is given by:

$$(\forall g\in\mathcal{G},\forall S\in\mathcal{M}')\ \ g\,I'\,S\ \textbf{iff}\ (\forall m\in S)\,g\,I\,m$$

The **row clarification** of $(\mathcal{G},\mathcal{M},I)$ is defined analogously. The **clarification** of $(\mathcal{G},\mathcal{M},I)$ is then given by the column clarification of the row clarification of $(\mathcal{G},\mathcal{M},I)$.

Example 3.9. Fig. 3.2 depicts the context resulting from : column clarification **(left)**, row clarification **(center)** and clarification **(right)** of the formal context depicted in Fig. 3.1. □

It should be noticed that for any formal context and after any type of clarification the concept lattice of the clarified context is order-isomorphic to the concept lattice of the base formal context. Proposition 3.2 shows that if $\mathbb{K}'$ is the column-clarification of $\mathbb{K}$ then $\mathbb{K}'_{ext}$ and $\mathbb{K}_{ext}$ are equal. Indeed, dual results can be obtained for the row-clarification and the set of intents.

Proposition 3.2. Let $\mathbb{K}$ be a context which column-clarification is $\mathbb{K}'$. We have: $\mathbb{K}'_{ext}=\mathbb{K}_{ext}$.

Proof. Recall that $\mathbb{K}_{ext}$ and $\mathbb{K}'_{ext}$ are closed under arbitrary intersection. We prove the double inclusion:

- $\mathbb{K}'_{ext} \subseteq \mathbb{K}_{ext}$: Let $A \in \mathbb{K}'_{ext}$, that is $\exists \mathbb{S} \subseteq \mathcal{M}'$ s.t. $ext_{\mathbb{K}'}(\mathbb{S}) = A$, that is: $A = \bigcap_{S \in \mathbb{S}} ext_{\mathbb{K}'}(\{S\})$. However, it is clear that: $ext_{\mathbb{K}'}(\{S\}) = \{g \in \mathcal{G} \mid g\, I'\, S\} = \{g \in \mathcal{G} \mid (\forall m \in S)\, g\, I\, m\} = ext_{\mathbb{P}}(S)$. Hence, $A = \bigcap_{S \in \mathbb{S}} ext_{\mathbb{K}}(S)$. In other words, $A \in \mathbb{K}_{ext}$ since it is the intersection of some elements in $\mathbb{K}_{ext}$.
- $\mathbb{K}_{ext} \subseteq \mathbb{K}'_{ext}$: Let $A \in \mathbb{K}_{ext}$, that is: $\exists B \subseteq \mathcal{M}$ s.t. $ext_{\mathbb{P}}(B) = A$, that is: $A = \bigcap_{m \in B} ext_{\mathbb{K}}(\{m\})$. For $m \in B$, let $S_m \in \mathcal{M}'$ be the unique set containing m. We have $ext_{\mathbb{K}'}(\{S_m\}) = ext_{\mathbb{K}}(\{m\})$. Hence, $A = \bigcap_{m \in B} ext_{\mathbb{K}'}(\{S_m\})$; that is $A \in \mathbb{K}'_{ext}$ since it is an intersection of some elements in $\mathbb{K}'_{ext}$.

This conclude the proof. ∎

Note 3.5. Further reduction of the context can be operated for a certain class of formal context (i.e. finite or doubly-founded ones). This can be done by removing also attributes/objects that can be seen as conjunction of other attributes/objects of the context. See [80] for more details. □

3.2.3 Handling Complex Attributes

Real world datasets do not come as formal contexts, i.e. all attributes are Boolean. Therefore, since (basic) FCA gives a tool to analyze datasets in a form of formal context, datasets with more complex attributes (eg. numerical or nominal attributes) need to be transformed to such a form before any manipulation. Such a transformation is called *conceptual scaling* (i.e. *binarizing*) [79]. In this section $(\mathcal{G}, \mathcal{M})$ denotes a dataset following Definition 3.1.

In order to perform a **conceptual scaling** on a dataset someone need to tell how each value of each attribute will be transformed to one or many boolean values. This is done using a **scale**.

Definition 3.11. Let m be an attribute which range of values is $\mathcal{R}_m$. A **scale** on m is any formal context $(\mathcal{R}_m, \mathcal{B}_m, I_m)$ where $\mathcal{B}_m$ is an arbitrary set and I_m is a binary relation.

The **nominal scale** of an attribute m is the formal context $(\mathcal{R}_m, \mathcal{R}_m, =)$.

Example 3.10. Fig. 3.3 **(center)** depicts the **nominal scale** of the attribute *"hair color"* given by $\big(\{brown, blond, black\}, \{brown, blond, black\}, =\big)$. □

Once a **scale** is defined on an attribute, one can perform a **(conceptual) scaling** in order to get the associated formal context.

Definition 3.12. The **(conceptual) scaling** of a dataset $(\mathcal{G}, \{m\})$ using the scale $(\mathcal{R}_m, \mathcal{B}_m, I_m)$ is the formal context $(\mathcal{G}, \mathcal{B}_m, I)$ where:

$$\big(\forall (g, b) \in \mathcal{G} \times \mathcal{B}_m\big)\ \ g\, I\, b \iff m(g)\, I_m\, b$$

The scaling of a dataset $(\mathcal{G}, \mathcal{M})$ with more than one attribute is simply the **apposition** (i.e. concatenation) of the contexts resulting from the scaling of each attribute $m \in \mathcal{M}$.

$\mathcal{G}$	hair color
g_1	brown
g_2	blond
g_3	brown
g_4	black

	brown	blond	black
brown	×		
blond		×	
black			×

$\mathcal{G}$	brown	blond	black
g_1	×		
g_2		×	
g_3	×		
g_4			×

Figure 3.3: **(Left)** a dataset with a categorical attribute *"hair color"*. **(Center)** A *nominal scale* on *"hair color"* values. **(Right)** The conceptual scaling using the nominal scale on *"hair color"*.

Example 3.11. Fig. 3.3 **(right)** depicts the conceptual scaling of the dataset presented left-side using the nominal scale presented in center. □

We have presented beforehand the most basic scale which is the **nominal scale**. Other types of scales exist and are thoroughly discussed in [80]. We present below some usual scales.

3.2.3.1 Discretization scalings

A usual scaling on a numerical attribute is *discretization*. Fig. 3.4 gives an example of a numerical dataset where the attribute age is discretizated into two buckets: $[0,30)$ representing young people and $[30,+\infty)$ representing the others.

$\mathcal{G}$	age
g_1	16
g_2	20
g_3	31
g_4	47

	$age \in [0,30)$	$age \in [30,+\infty)$
16	×	
20	×	
31		×
47		×

$\mathcal{G}$	$age \in [0,30)$	$age \in [30,+\infty)$
g_1	×	
g_2	×	
g_3		×
g_4		×

Figure 3.4: **(Left)** a dataset with a numerical attribute "age". **(Center)** A scale for discretization. **(Right)** The resulting conceptual scaled context.

3.2.3.2 Interordinal scalings

Another, more expressive, scaling for numerical attributes is *interordinal scaling*. Fig. 3.5 presents the interordinal scaling of the dataset depicted in Fig. 3.4 **(left)**. It is clear that itemsets in an interordinal scaled dataset represent interval restrictions on the attribute (see [104]).

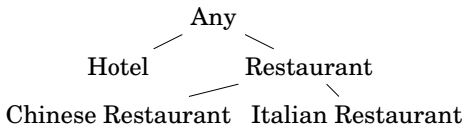
$\mathcal{G}$	$age \leq 16$	$age \leq 20$	$age \leq 31$	$age \leq 47$	$age \geq 16$	$age \geq 20$	$age \geq 31$	$age \geq 47$
g_1	×	×	×	×	×			
g_2		×	×	×	×	×		
g_3			×	×	×	×	×	
g_4				×	×	×	×	×

Figure 3.5: Interordinal scaled context of the dataset depicted in Fig. 3.4 (left).

3.2.3.3 Ordinal scalings

Such a type of scaling is useful when dealing with datasets where there is a hierarchy between items. Consider for instance the dataset depicted in Fig. 3.6 **(top-left)** where objects are places and each place is tagged. The tags are organized under a taxonomy, i.e. a hierarchy presented in the Fig. 3.6 **(top-right)**. This taxonomy can be seen as a poset $(T, \leq)$ where T represents the set of all possible tags (i.e. $T = \{Any, Hotel, ...\}$) and $\leq$ represents the order *"is a specialization of"*. For instance, *Italian Restaurant* $\leq$ *Restaurant*. The **ordinal scale** is then simply given by $(T, T, \leq)$. The *ordinal scaled* context is depicted in Fig. 3.6 **(bottom)**.

$\mathcal{G}$	tag
$place_1$	Hotel
$place_2$	Chinese Restaurant
$place_3$	Italian Restaurant



$\mathcal{G}$	Any	Hotel	Restaurant	Chinese Restaurant	Italian Restaurant
$place_1$	×	×			
$place_2$	×		×	×	
$place_3$	×		×		×

Figure 3.6: (**top-left**) Point-of-interests annotated with tags. **(top-right)** a taxonomy of tags. **(bottom)** an ordinal-scaled context

3.2.4 Discussion

Even if Formal Concept Analysis provides a broad framework to handle binary datasets or datasets with more complex attributes thanks to conceptual scaling. Binarizing a dataset with regard to the patterns we want to look for could be sometime tricky.

In response to that, a more natural way to handle complex datasets was introduced in [78] under the name of **pattern structures** (see Section 3.4). Objects in a pattern structure have descriptions (e.g. the equivalent notion to itemsets in $\wp(\mathcal{M})$ in a formal context) with a meet-semilattice operation on them (e.g. equivalent to set intersection in $(\wp(\mathcal{M}), \subseteq)$ in a formal context). This framework proved its usefulness in many data analysis tasks (see [112]).

However, pattern structures require the *description space* to be a (upper-bounded) meet-semilattice which is not the case for all description spaces such as sequence of itemsets patterns [48]. **Pattern setups** were introduced in [123] to generalize pattern structures by demanding only a partial order on descriptions. This latter framework will be studied in the next section.

3.3 Pattern Setups

Pattern setups model pattern search spaces simply as partially ordered sets. This makes them a very broad framework as they do not require additional properties on the pattern languages. The term *"pattern setup"* and its definition was proposed in [123]. Later, we proposed in [18] a more detailed investigation on this new framework in the aim of better understanding it. This section presents the basic notions related to pattern setups introduced in [18].

3.3.1 Basic Definitions

Let us start by defining what is a description space in general.

Definition 3.13. A **description space**; also called **description language**, **pattern space** or **pattern language**; is any *poset* $\underline{\mathcal{D}} := (\mathcal{D}, \sqsubseteq)$. Elements of $\mathcal{D}$ are called **descriptions** or **patterns**. For any $c, d \in \mathcal{D}$, $c \sqsubseteq d$ should be read as "c is *less restrictive than* d" or "c **subsumes** d".

We define below the notion of pattern setup following [123].

Definition 3.14. A **pattern setup** is a triple $\mathbb{P} = (\mathcal{G}, \underline{\mathcal{D}}, \delta)$ where $\mathcal{G}$ is a *set* (of objects), $\underline{\mathcal{D}}$ is a description space and $\delta : \mathcal{G} \to \mathcal{D}$ defines a mapping that takes each object $g \in \mathcal{G}$ to its description $\delta(g) \in \mathcal{D}$. Let $g \in \mathcal{G}$ and $d \in \mathcal{D}$ be an object and a description, respectively. We say that object g **realizes** description d or description d **hold for** or **cover** object g iff $d \sqsubseteq \delta(g)$.

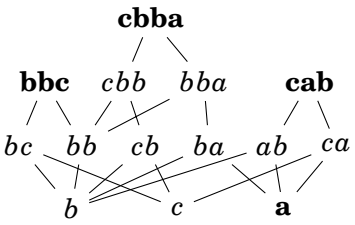
We present below two examples of pattern setups.

Example 3.12. Consider the pattern setup $\mathbb{P} = (\mathcal{G}, \underline{\mathcal{D}}, \delta)$ in Fig. 3.7. We have $\mathcal{G} = \{g_i\}_{1 \leq i \leq 4}$. The description space is the set of *nonempty words* on the alphabet $\{a, b, c\}$ (i.e. $\{a, b, c\}^+$) ordered by the relationship *"is substring of"* $\sqsubseteq$. The mapping δ associates to each objects in $\mathcal{G}$ its word in the description space. For instance $\delta(g_1) =$ "cab". The diagram in the center of Fig. 3.7 depicts the Hasse Diagram of the poset $(\downarrow \delta[\mathcal{G}], \sqsubseteq)$ with $\delta[\mathcal{G}] =$ {"**cab**","**cbba**","**a**","**bbc**"}. In other words, it depicts the set of descriptions $d \in \mathcal{D}$ holding for at least one object in $\mathcal{G}$. It is clear that the description "ca" holds for g_1 since "ca" $\sqsubseteq$ "cab". However, description "cb" does not hold for g_1 since "cb" is not a substring of "cab". More generally, descriptions holding for g_i is the principal filter of $\delta(g_i)$ (i.e. $\downarrow \delta(g_i)$). For instance, the set of descriptions holding for g_1 is given by $\downarrow \delta(g_1) =$ {"a","ca","ab","**cab**"}. □

Example 3.13. Consider the *pattern setup* $\mathbb{P} = (\mathcal{G}, \underline{\mathcal{D}}, \delta)$ presented in Fig. 3.9 (left). The set of objects is $\mathcal{G} = \{g_i\}_{1 \leq i \leq 4}$ and the description space $\underline{\mathcal{D}}$ is the powerset ordered by set inclusion $(\wp(\mathcal{M}), \subseteq)$ (i.e. itemsets) with $\mathcal{M} = \{a, b, c\}$. Again descriptions holding for g_4 are all itemsets included in $\delta(g_4) = \{b, c\}$ (i.e. $\downarrow \delta(g_4) = \{\emptyset, \{b\}, \{c\}, \{b, c\}\}$). □

We have seen that the relation **"realizes"** builds a binary relation between objects and descriptions (see Definition 3.14). Based on this binary relation, two key operators, namely **extent** and **cover** are derived (see Definition 3.15 and 3.16).

$\mathcal{G}$	$\delta(\cdot)$
g_1	"**cab**"
g_2	"**cbba**"
g_3	"**a**"
g_4	"**bbc**"



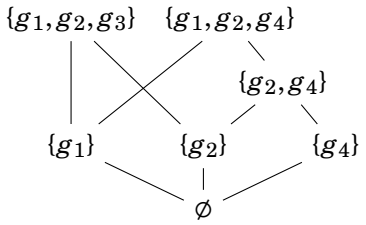


Figure 3.7: The table (left) represents the mapping function δ of the pattern setup considered in example 3.12. The diagram (center) represents the set of substrings in $\{a,b,c\}^+$ holding for at least one object in $\mathcal{G}$. The diagram (right) represents the poset of definable sets $(\mathbb{P}_{ext},\subseteq)$.

Definition 3.15. The **extent operator**, denoted by ext, is the operator that takes each description $d \in \mathcal{D}$ to the subset of objects in $\mathcal{G}$ *realizing* it:

$$ext : \mathcal{D} \to \wp(\mathcal{G}), d \mapsto \{g \in \mathcal{G} \mid d \sqsubseteq \delta(g)\}$$

The size of $ext(d)$ is called the **support** of d, i.e. $support : d \mapsto |ext(d)|$.

Note 3.6. Please note that for any $S \subseteq \mathcal{D}$, we denote $ext[S] = \{ext(d) \mid d \in S\}$. □

Definition 3.16. The **cover operator**, denoted by cov, takes each subset of objects $A \subseteq \mathcal{G}$ to the set of common descriptions in $\mathcal{D}$ *covering* all of them:

$$cov : \wp(\mathcal{G}) \to \wp(\mathcal{D}), A \mapsto \delta[A]^{\ell} = \bigcap_{g \in A} \downarrow \delta(g) = \{d \in \mathcal{D} \mid (\forall g \in A)\, d \sqsubseteq \delta(g)\}$$

Example 3.14. Consider the pattern setup presented in Example 3.12. We have: $ext(\text{“}bb\text{”}) = \{g_2, g_4\}$ and $cov(\{g_2, g_4\}) = \downarrow \delta(g_2) \cap \downarrow \delta(g_4) = \{\text{“}b\text{”}, \text{“}bb\text{”}, \text{“}c\text{”}\}$. □

Definition 3.17. A subset $A \subseteq \mathcal{G}$ is said to be:

- **Definable**, **Separable** or an **Extent** if there exists at least one description $d \in \mathcal{D}$ isolating A from the other objects in $\mathcal{G}$, i.e. $A = ext(d)$.
- **Coverable** if objects in A share at least one common description, i.e. $cov(A) \neq \emptyset$.

The set of **definable sets** is denoted $\mathbb{P}_{ext}$ and given by: $\mathbb{P}_{ext} := ext[\mathcal{D}] = \{ext(d) \mid d \in \mathcal{D}\}$.

Please note that poset $(\mathbb{P}_{ext},\subseteq)$ forms a subposet of $(\wp(\mathcal{G}),\subseteq)$, that is definable sets are naturally ordered by $\subseteq$. The set of *coverable sets* is naturally given by $\downarrow \mathbb{P}_{ext}$. In other words, any subset of a *coverable set* is *coverable*. Conversely, any superset of a *non-coverable set* is a *non-coverable set*.

Example 3.15. The poset of definable sets $(\mathbb{P}_{ext},\subseteq)$ associated to the pattern setup presented in Example 3.12 is depicted in Fig. 3.7 (right). It is clear that $\{g_2, g_4\}$ is definable since $ext(\text{“}bb\text{”}) = \{g_2, g_4\}$. However, there is no description which extent is exactly $\{g_1, g_2\}$, hence $\{g_1, g_2\}$ will be said **non-definable**. Still, $\{g_1, g_2\}$ is coverable since g_1 and g_2 share at least one common descritpion (i.e. $cov(\{g_1, g_2\} = \{\text{“}a\text{”}, \text{“}b\text{”}, \text{“}c\text{”}\} \neq \emptyset$). One should note also that $\{g_3, g_4\}$ is **non-coverable** since they share no common symbol and the empty string is excluded from the pattern space. □

An important property arising directly from the definition of both ext and cov is given in Proposition 3.3. It tells that: on the one hand, the more restrictive is a description, the less it covers objects in the database. On the other hand, the bigger is a set of objects the less its elements share descriptions in common.

Proposition 3.3. Operators ext and cov are *order-reversing*:

$$(\forall c,d \in \mathcal{D})\ c \sqsubseteq d \Rightarrow ext(d) \subseteq ext(c) \qquad (\forall A,B \subseteq \mathcal{G})\ A \subseteq B \Rightarrow cov(B) \subseteq cov(A)$$

Proof. We have:

1. Let $A \subseteq B \subseteq \mathcal{G}$, let $d \in cov(B)$, thus $(\forall g \in B)\, d \sqsubseteq \delta(g)$. Since $A \subseteq B$, we conclude that $(\forall g \in A)\, d \sqsubseteq \delta(g)$ that is $d \in cov(A)$. Thus $cov(B) \subseteq cov(A)$.
2. Let $c,d \in \mathcal{D}$ such that $c \sqsubseteq d$. Let $g \in ext(d)$, that is $d \sqsubseteq \delta(g)$ thus $c \sqsubseteq \delta(g)$; that is $g \in ext(c)$. We conclude that $ext(d) \subseteq ext(c)$.

This concludes the proof. ∎

We have seen in Proposition 3.3 that mappings ext and cov are order reversing. Hence, one could think that (ext, cov) may form some Galois connection. However, it is not the case since ext associates an extent to one description while cov outputs a set of descriptions rather than one. In other words, these two mappings are not compatibles. Yet, we will see in next section that ext will be involved into a Galois connection when the considered pattern setup verifies additional properties (i.e. the pattern setup is a pattern structure [78]). Mapping cov will also be involved in another Galois connection in Section 3.7.

Definition 3.18. For $c,d \in \mathcal{D}$, the **pattern implication** $c \to d$ holds if $ext(c) \subseteq ext(d)$. That is, every object realizing c realizes d. Dually, for $A,B \subseteq \mathcal{G}$, the **object implication** $A \to B$ holds if $cov(A) \subseteq cov(B)$. That is, every description covering all object in A covers also all objects in B.

We say that $c,d \in \mathcal{D}$ are **equivalent** if $c \to d$ and $d \to c$ and we have $ext(c) = ext(d)$. Dual definition can be given for the *equivalence between object sets*.

It is clear that if $d \sqsubseteq c$ and since ext is an *order-reversing mapping*, we have $c \to d$. Regarding this observation, there is two types of implications between descriptions: (i) implications deduced directly from $\sqsubseteq$ and (ii) implications that are dependent on the pattern setup. While the former implications are intrinsic to the description space, the latter are more **informative** since they are those enclosing the knowledge hidden in the pattern setup.

Example 3.16. In the pattern setup presented in Example 3.12 and Fig. 3.7, we have $ext(\text{“}bb\text{”}) = \{g_2, g_4\}$ and $ext(\text{“}c\text{”}) = \{g_1, g_2, g_4\}$. Hence, we have “$bb$” → “$c$” or in other words in every string containing “bb” in the pattern setup contains also “c”. □

Proposition 3.4 gives characterizations of the set $ext[cov(A)]$ and $cov(ext(d))$ for $A \subseteq \mathcal{G}$ and $d \in \mathcal{D}$. This proposition will be useful later in this chapter.

Proposition 3.4. For $A \subseteq \mathcal{G}$ and $d \in \mathcal{D}$:

$$\begin{aligned} ext[cov(A)] &= \{E \in \mathbb{P}_{ext} \mid A \subseteq E\} = \uparrow A \cap \mathbb{P}_{ext} \\ cov(ext(d)) &= \{c \in \mathcal{D} \mid ext(d) \subseteq ext(c)\} = \{c \in \mathcal{D} \mid d \rightarrow c\} \end{aligned}$$

Proof. We show the two equations separately:

(1) Let $E \subseteq \mathcal{G}$, we have: $E \in ext[cov(A)] \Leftrightarrow (\exists d \in cov(A))\, E = ext(d) \Leftrightarrow (\exists d \in \mathcal{D} \forall g \in A)\, d \sqsubseteq \delta(g) \Leftrightarrow (\exists d \in \mathcal{D})\, A \subseteq ext(d) = E \Leftrightarrow E \in \mathbb{P}_{ext} \cap \uparrow A$. Therefore, $ext[cov(A)] = \{E \in \mathbb{P}_{ext} \mid A \subseteq E\} = \uparrow A \cap \mathbb{P}_{ext}$.

(2) Let $c \in \mathcal{D}$, we have: $c \in cov(ext(d)) \Leftrightarrow (\forall g \in ext(d))\, c \sqsubseteq \delta(g) \Leftrightarrow (\forall g \in ext(d))\, g \in ext(c) \Leftrightarrow ext(d) \subseteq ext(c)$. We conclude that $cov(ext(d)) = \{c \in \mathcal{D} \mid ext(d) \subseteq ext(c)\} = \{c \in \mathcal{D} \mid c \rightarrow d\}$. ∎

Example 3.17. Consider the pattern setup presented in Example 3.12 and its associated $(\mathbb{P}_{ext}, \subseteq)$ depicted in Fig. 3.7 (right). We have: $ext(\text{“}bb\text{”}) = \{g_2, g_4\}$ and $cov(\{g_2, g_4\}) = \{\text{“}b\text{”}, \text{“}bb\text{”}, \text{“}c\text{”}\}$. Hence:

- $ext[cov(\{g_2, g_4\})] = \{ext(\text{“}b\text{”}), ext(\text{“}bb\text{”}), ext(\text{“}c\text{”})\} = \{\{g_1, g_2, g_4\}, \{g_2, g_4\}\}$.
- $cov(ext(\text{“}bb\text{”})] = cov(\{g_2, g_4\}) = \{\text{“}b\text{”}, \text{“}bb\text{”}, \text{“}c\text{”}\}$. □

3.3.2 A Minimal Representation of a Pattern Setup

An important notion analogous to what is called *representation context* in pattern structures (see [41, 78]) is introduced in Theorem 3.1. Technically, such a representation does not provide a practical way to explore definable sets of an arbitrary pattern setups, but helps to simulate definable sets search space of a pattern setup independently from the description space. Before introducing the Theorem, we shall state Lemma 3.1 and Proposition 3.5.

Lemma 3.1. For any object $g \in \mathcal{G}$, the smallest definable set in $\mathbb{P}_{ext}$ enclosing it is given by $ext(\delta(g))$. Formally:

$$(\forall g \in \mathcal{G})\ ext(\delta(g)) = \bigcap (\uparrow \{g\} \cap \mathbb{P}_{ext})$$

Proof. Recall that $ext[cov(\{g\})] = \uparrow \{g\} \cap \mathbb{P}_{ext}$ (See proposition 3.4). Let $g \in \mathcal{G}$, we have $\delta(g) \in cov(\{g\})$, thus $ext(\delta(g)) \in ext[cov(\{g\})]$. Let us show that $ext(\delta(g))$ is a lower bound of $ext[cov(\{g\})]$. We have: $cov(\{g\}) = \{d \in \mathcal{D} \mid d \sqsubseteq \delta(g)\}$. Thus, $\forall d \in cov(\{g\}) : d \sqsubseteq \delta(g)$. Since ext is an order reversing operator, we obtain: $\forall A \in ext[cov(\{g\})] : ext(\delta(g)) \subseteq A$. Thus, $ext(\delta(g))$ is the smallest element of $ext[cov(\{g\})]$. That is $\bigcap (\uparrow \{g\} \cap \mathbb{P}_{ext}) = ext(\delta(g))$. ∎

Proposition 3.5. Let $\mathcal{G}$ be a non empty finite set and let $S \subseteq \wp(\mathcal{G})$. We have

$$\exists \mathbb{P} \textit{ a pattern setup such that } S = \mathbb{P}_{ext} \iff (\forall g \in \mathcal{G}) \bigcap (\uparrow \{g\} \cap S) \in S$$

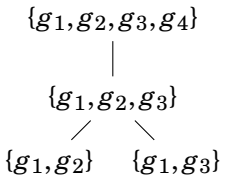


Figure 3.8: A family $S \subseteq \wp(\{g_1, g_2, g_3, g_4\})$ for which there is no pattern setup $\mathbb{P}$ s.t. $\mathbb{P}_{ext} = S$

Proof. We show now the two implications independently:

($\Rightarrow$) Let $S = \mathbb{P}_{ext}$ for some pattern setup $\mathbb{P}$. Using Lemma 3.1, $\bigcap(\uparrow \{g\} \cap S) \in S$.

($\Leftarrow$) Let $S \subseteq \wp(\mathcal{G})$ for which $\forall g \in \mathcal{G}$ we have $\bigcap(\uparrow \{g\} \cap S) \in S$. Let us now define the following pattern setup:

$$\mathbb{P} = \left(\mathcal{G}, (S, \supseteq), \delta : g \mapsto \bigcap(\uparrow g \cap S)\right)$$

Let $A \in S$ be a description, we have: $ext(A) = \{g \in \mathcal{G} \mid \bigcap(\uparrow \{g\} \cap S) \subseteq A\}$. Let us show that $ext(A) = A$ by showing double inclusion: **(1)** Let $g \in A$, thus $A \in (\uparrow \{g\} \cap S)$. It follows that $\bigcap(\uparrow \{g\} \cap S) \subseteq A$. We conclude that $g \in ext(A)$. Therefor $A \subseteq ext(A)$. **(2)** Let $g \in ext(A)$, thus $\bigcap(\uparrow \{g\} \cap S) \subseteq A$. Since $\forall B \in \uparrow \{g\} \cap S$ we have $g \in B$, we have $g \in \bigcap(\uparrow \{g\} \cap S)$, that is $g \in A$. We conclude that $ext(A) \subseteq A$. Both inclusion leads us to have $(\forall A \in S)\, ext(A) = A$. We conclude that $ext[S] = S$. In other words, $\mathbb{P}_{ext} = S$.

This concludes the proof. ∎

Proposition 3.5 tells that not all families of subsets of $\mathcal{G}$ could be seen as a set of extents of some pattern setup. Example 3.18 and Fig. 3.8 illustrate this claim.

Example 3.18. Consider the poset depicted in Fig. 3.8 where $\mathcal{G} = \{g_1, g_2, g_3, g_4\}$ and $S = \{\{g_1, g_2\}, \{g_1, g_3\}, \{g_1, g_2, g_3\}, \{g_1, g_2, g_3, g_4\}\}$ can never be seen as a $\mathbb{P}_{ext}$ for some pattern setup $\mathbb{P}$. Indeed, $\uparrow \{g_1\} \cap S = \{\{g_1, g_2\}, \{g_1, g_3\}\}$ whose intersection is not in S. □

Theorem 3.1. For any pattern setup $\mathbb{P}$, the pattern setup $\mathbb{R}(\mathbb{P})$ given by

$$\mathbb{R}(\mathbb{P}) = \left(\mathcal{G}, (\mathbb{P}_{ext}, \supseteq), g \mapsto \bigcap(\uparrow \{g\} \cap \mathbb{P}_{ext})\right)$$

is called the **minimal representation** of $\mathbb{P}$ and we have $\mathbb{R}(\mathbb{P})_{ext} = \mathbb{P}_{ext}$.

Proof. Theorem 3.1 is a corollary of Proposition 3.5. Indeed, the pattern setup $\mathbb{R}(\mathbb{P})$ is the same as the one built in the proof of Proposition 3.5 ($\Leftarrow$) since $\mathbb{P}_{ext}$ is a set system verifying the property $(\forall g \in \mathcal{G}) \bigcap(\uparrow \{g\} \cap \mathbb{P}_{ext}) \in \mathbb{P}_{ext}$ (i.e. implication ($\Rightarrow$)). We have $\mathbb{R}(\mathbb{P})_{ext} = \mathbb{P}_{ext}$. Moreover, this representation is said to be minimal since any proper subposet of $(\mathbb{P}_{ext}, \supseteq)$ will drop at least one definable set. ∎

3.4 Pattern Structures

Pattern structures were introduced in [78]. They require that every set of objects has a greatest common description. A formal definition is given in Definition 3.19. Pattern structures provide a very strong tool to formalize a large class of pattern languages [112]. For instance, pattern setups over the language of itemsets [4, 80], intervals [104], convex polygons [20], sequence sets [39][2], and graph sets[110][2] are all pattern structures.

Definition 3.19. A pattern setup $\mathbb{P} = (\mathcal{G}, \underline{\mathcal{D}}, \delta)$ is said to be a **pattern structure** iff every subset of objects has a greatest common description. Formally: $(\forall S \subseteq \delta[\mathcal{G}])\ S$ has a meet $\bigsqcap S$.

Example 3.19. The pattern setup presented in Example 3.13 and Fig. 3.9 is a pattern structure. Indeed, since the description space $(\mathcal{D}, \sqsubseteq)$ is the powerset lattice $(\wp(\{a,b,c\}), \subseteq)$ (i.e a complete lattice) then every subsets $S \subseteq \delta[\mathcal{G}] \subseteq \mathcal{D}$ does have a meet which is the set intersection $\bigcap S$.

However, the pattern setup $\mathbb{P}$ presented in Example 3.12 and Fig. 3.7 is not a pattern structure. Indeed, the set of common descriptions $cov(\{g_2, g_4\}) = \{\text{“}b'', \text{“}bb'', \text{“}c''\}$ does not have a maximum (i.e. $\{\delta(g_2), \delta(g_4)\}$ does not have a meet) since $\{\text{“}b'', \text{“}bb'', \text{“}c''\}$ has two maximal elements. □

Thanks to the existence of the meet, we define a new operator below.

Definition 3.20. The **intent operator**, denoted by int, takes each subset of objects $A \subseteq \mathcal{G}$ to the greatest common description in $\mathcal{D}$ *covering* them (i.e. the maximum of $cov(A)$). Formally:

$$int : \wp(\mathcal{G}) \to \mathcal{D}, A \mapsto inf\, \delta[A] = \bigsqcap \delta[A]$$

Interestingly, analogously to Proposition 3.1, we have the following property:

Proposition 3.6. Let $\mathbb{P}$ be a pattern structure, then the pair (ext, int) forms a *Galois connection* between posets $(\mathcal{D}, \sqsubseteq)$ and $(\wp(\mathcal{G}), \subseteq)$ (see Definition 2.25).

Proof. We have ext and cov are order-reversing (see proposition 3.3). Let $A, B \subseteq \mathcal{G}$ s.t. $A \subseteq B$, we have $cov(B) \subseteq cov(A)$. Since $int(B)$ is the maximum of $cov(B)$ and $int(A)$ is the maximum of $cov(A)$ then $int(B) \sqsubseteq int(A)$. Let us show that $ext \circ int$ and $int \circ ext$ are extensive:

- $ext \circ int$ **is extensive:** Let $A \subseteq \mathcal{G}$ and let us show that $A \subseteq ext(int(A))$. The case $A = \emptyset$ is trivial since the $\emptyset$ is the bottom element of $\wp(\mathcal{G})$. Let $A \neq \emptyset$ and $g \in A$, since $int(A) = \bigsqcap_{g \in A} \delta(g)$ then $int(A) \sqsubseteq \delta(g)$. Hence, by definition $g \in ext(int(A))$.
- $int \circ ext$ **is extensive:** Let $d \in \mathcal{D}$, we have $ext(d) = \{g \in \mathcal{G} \mid d \sqsubseteq \delta(g)\}$. Hence, $int(ext(d)) = \bigsqcap S$ where $S = \{\delta(g) \mid g \in \mathcal{G} \text{ and } d \sqsubseteq \delta(g)\}$. It is clear that $d \in S^{\ell}$ and since $int(ext(d)) = \bigsqcap S$ is the maximum of S^{ℓ} we obtain that $d \sqsubseteq int(ext(d))$.

This concludes the proof. ∎

[2]Sequence and graph patterns do not induce pattern structures directly, but the sets of incomparable patterns do.

According to Proposition. 3.6 and Theorem. 2.9, mappings $ext \circ int$ and $int \circ ext$ form *closure operators* on the two posets $(\wp(\mathcal{G}), \subseteq)$ and $(\mathcal{D}, \sqsubseteq)$ respectively. Thanks to this Galois Connection, one can define a complete lattice based on the closed elements.

Definition 3.21. The **(pattern) concept lattice** associated to the pattern structure $\mathbb{P} = (\mathcal{G}, \underline{\mathcal{D}}, \delta)$ is the *complete lattice* denoted by $\underline{\mathfrak{B}}(\mathbb{P}) = (\mathfrak{B}(\mathbb{P}), \leq)$. Elements of $\mathfrak{B}(\mathbb{P})$ are called **(pattern) concepts** and are given by:

$$(A, d) \in \wp(\mathcal{G}) \times \mathcal{D} \quad s.t. \quad A = ext(d) \; and \; d = int(A)$$

The concepts are ordered by $\leq$ as follows: $(A_1, d_1) \leq (A_2, d_2) \Leftrightarrow A_1 \subseteq A_2 \Leftrightarrow d_2 \sqsubseteq d_1$.

Two complete lattices isomorphic to the *concept lattice* can be derived:

1. The poset of *definable sets* $(\mathbb{P}_{ext}, \subseteq)$ which is a closed under intersection. Note that *definable sets* are the fixpoints of the closure operator $ext \circ int$.
2. The poset of *closed patterns* $(\mathbb{P}_{int}, \sqsupseteq)$ with $\mathbb{P}_{int} = int[\wp(\mathcal{G})] = \{\bigsqcap \delta[A] \mid A \subseteq \mathcal{G}\}$ is a complete lattice. Elements of $\mathbb{P}_{int}$ are called *closed patterns* since they are fixpoints of the closure operator $int \circ ext$.

Example 3.20. Consider again the *pattern structure* $\mathbb{P} = (\mathcal{G}, (\wp(\{a, b, c\}), \subseteq), \delta)$ presented in Fig. 3.9. Since the meet is the set intersection we have: $int(\{g_1, g_4\}) = \delta(g_1) \cap \delta(g_4) = \{a, b, c\} \cap \{b, c\} = \{b, c\}$.

Concept lattice $\underline{\mathcal{B}}(\mathbb{P})$ is depicted in Fig. 3.9 (right). Note that $\mathbb{P}_{ext}$ (resp. $\mathbb{P}_{int}$) can be obtained directly from the concept lattice by taking the extent (resp. intent) of each pattern concept. □

We draw the reader attention to the fact that the closure operator $ext \circ int$ takes each subset of object $A \subseteq \mathcal{G}$ to the smallest definable set $ext \circ int(A) \in \mathbb{P}_{ext}$ enclosing it. This observation is formally stated in the proposition below.

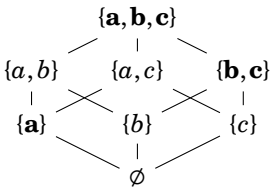
Proposition 3.7. We have:

$$ext \circ int : \wp(\mathcal{G}) \to \wp(\mathcal{G}), A \mapsto \bigcap \{E \in \mathbb{P}_{ext} \mid A \subseteq E\} = \bigcap (\uparrow A \cap \mathbb{P}_{ext})$$

Proof. This result is straightforward from the fact that $ext \circ int$ is a closure operator. Indeed, according to Theorem 1 in [80] (page 8), we have $\mathbb{P}_{ext} = \{ext \circ int(A) \mid A \subseteq \mathcal{G}\}$ is closure system. By application of the theorem we have: $ext \circ int : A \mapsto \bigcap \{E \in \mathbb{P}_{ext} \mid A \subseteq E\}$.

It is important to highlight the fact that for any formal context $(\mathcal{G}, \mathcal{M}, I)$, the pattern structure $\mathbb{P} = (\mathcal{G}, (\wp(\mathcal{M}), \subseteq), \delta)$ where $\delta : g \mapsto \{m \in \mathcal{M} \mid g \, I \, m\}$ hold exactly the same concepts, i.e. $\underline{\mathfrak{B}}(\mathcal{G}, \mathcal{M}, I) = \underline{\mathfrak{B}}(\mathbb{P})$. Interestingly, the converse holds also, that is any pattern structure $\mathbb{P}$ can be seen as a formal context, called **representation context**. The representation context is built over the same set of objects and have the same set of extents $\mathbb{P}_{ext}$. For more details about how this context is built, please read [41, 78].

$\mathcal{G}$	$\delta(\cdot)$
g_1	$\{\mathbf{a},\mathbf{b},\mathbf{c}\}$
g_2	$\{\mathbf{a}\}$
g_3	$\{\mathbf{a}\}$
g_4	$\{\mathbf{b},\mathbf{c}\}$



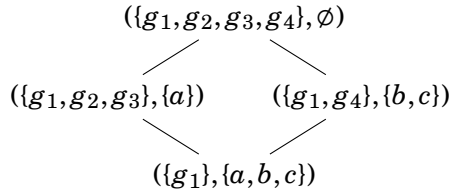


Figure 3.9: The table (left) represents the pattern setup $\mathbb{P} = \left(\mathcal{G}, \underline{\mathcal{D}}, \delta\right)$ with $\mathcal{G} = \{g_i\}_{1\leq i\leq 4}$, $\underline{\mathcal{D}} = (\wp(\{a,b,c\}), \subseteq)$ depicted by the Hasse Diagram (center) and δ maps an object to its itemset. The diagram (right) depicts the concept lattice $\underline{\mathcal{B}}(\mathbb{P})$.

One could revisit the definition of implication between set of objects given in Definition 3.18 in a pattern structure. Indeed, for any subset of objects $A, B \in \wp(\mathcal{G})$, we have $cov(A) \subseteq cov(B)$ *iff* $int(A) \sqsubseteq int(B)$. Hence, $A \to B$ *iff* $int(A) \sqsubseteq int(B)$. Proposition 3.8 states the most informative implications for a given premise.

Proposition 3.8. Let $d \in \mathcal{D}$ and $A \subseteq \mathcal{G}$, we have:

$$(\forall c \in \mathcal{D})\left(d \to c \textbf{ iff } c \sqsubseteq int(ext(d))\right) \textbf{ and } (\forall B \subseteq \mathcal{G})\left(A \to B \textbf{ iff } B \subseteq ext(int(A))\right)$$

Proof. Let $d \in \mathcal{D}$. Let us show that $(\forall c \in \mathcal{D})\left(d \to c \textbf{ iff } c \sqsubseteq int(ext(d))\right)$:

- ($\Leftarrow$) Let $c \in \mathcal{D}$ s.t. $c \sqsubseteq int(ext(d))$. Since ext is order-reversing, we have $ext(int(ext(d))) \subseteq ext(c)$. Moreover, since (ext, int) form a Galois connection, we have $ext(int(ext(d))) = ext(d)$. Therefore, $ext(d) \subseteq ext(c)$ or in other words $d \to c$.
- ($\Rightarrow$) Let $c \in \mathcal{D}$ s.t. $d \to c$, we have $ext(d) \subseteq ext(c)$. Hence, $int(ext(c)) \sqsubseteq int(ext(d))$. Since $int \circ ext$ is extensive, we have $c \sqsubseteq int(ext(c)) \sqsubseteq int(ext(d))$.

Similar proof can be done for the second part of the proposition. ∎

Definition 3.19 presents pattern structures following [123]. The original equivalent definition of pattern structure [78] requires that the description space $(\mathcal{D}, \sqsubseteq)$ must be a meet-semilattice for which $\left\{\sqcap \delta[S] \mid S \subseteq \mathcal{G}\right\}$ forms a complete subsemilattice of $(\mathcal{D}, \sqsubseteq)$. Theorem 3.2 builds a bridge between meet-semilattices and pattern structures over finite set of objects or more generally between pattern structures and complete lattices.

Theorem 3.2. Let $\underline{\mathcal{D}} = (\mathcal{D}, \sqsubseteq)$ be a poset, the following properties are equivalent:

- For any finite set $\mathcal{G} \neq \emptyset$ and any $\delta \in \mathcal{D}^{\mathcal{G}}$, $(\mathcal{G}, \underline{\mathcal{D}}, \delta)$ is a pattern structure[3].
- $\underline{\mathcal{D}}$ is a an upper-bounded *meet-semilattice* (i.e. $\emptyset$ has also a *meet*).

More generally, the following properties are equivalent:

- For any set $\mathcal{G} \neq \emptyset$ and any $\delta \in \mathcal{D}^{\mathcal{G}}$, $(\mathcal{G}, \underline{\mathcal{D}}, \delta)$ is a pattern structure.
- $\underline{\mathcal{D}}$ is a *complete lattice*.

[3]Set $\mathcal{D}^{\mathcal{G}}$ denotes the set of all mappings $\delta : \mathcal{G} \to \mathcal{D}$

Proof. Let us show both implications for a finite $\mathcal{G}$:

$\Rightarrow$ The empty set has a meet in $\underline{\mathcal{D}}$ since $\delta[\emptyset] = \emptyset$ has a meet. Thus $\underline{\mathcal{D}}$ has a top element $\top = \bigsqcup \mathcal{D} = \bigsqcap \emptyset$. Moreover, let $S \subseteq \mathcal{D}$ be a finite set, one can build a finite set $\mathcal{G}$ such that $\delta[\mathcal{G}] = S$. Since $\mathbb{P}$ is a pattern structure then $S = \delta[\mathcal{G}]$ has a meet. We conclude that $\underline{\mathcal{D}}$ is an upper-bounded meet-semilattice.

$\Leftarrow$ Let $\mathbb{P} = (\mathcal{G}, \underline{\mathcal{D}}, \delta)$ be a pattern setup. Any subset of $\delta[\mathcal{G}]$ is finite subset of $\mathcal{D}$ and thus has a meet (including the $\emptyset$ since $\mathcal{D}$ has its top element).

Let us now consider the case of arbitrary set $\mathcal{G}$:

$\Rightarrow$ Let $S \subseteq \mathcal{D}$, one can build $\mathcal{G}$ such that $\delta[\mathcal{G}] = S$. Since $\mathbb{P}$ is a pattern structure then $\delta[\mathcal{G}] = S$ has a meet. We conclude that $\underline{\mathcal{D}}$ is a complete lattice.

$\Leftarrow$ Let $\mathbb{P} = (\mathcal{G}, \underline{\mathcal{D}}, \delta)$ be a pattern setup. Any subset of $\delta[\mathcal{G}]$ is a subset of $\mathcal{D}$ and thus has a meet (including the $\emptyset$ since $\mathcal{D}$ has its top element).

This concludes the proof. ∎

The state-of-the-art abounds with examples of descriptions spaces that are complete lattices that someone can use to build pattern structures:

- *Itemset pattern structure [80].* The description space is the Boolean lattice $(\wp(\mathcal{M}), \subseteq)$ where $\mathcal{M}$ is a non empty finite set of attributes.
- *Interval pattern structure [104].* The description space is the complete lattice of all axis-parallel m-dimensional hyperrectangles ordered by $\supseteq$. See Section 5.4 for more details.
- *Convex sets pattern structure [20].* The description space is the complete lattice of all convex sets in $\mathbb{R}^m$ ordered by inclusion $(\mathfrak{C}(\mathbb{R}^m), \supseteq)$[4]. See Section 5.5 for more details.
- *Partition pattern structure [49].* The description space is the complete lattice of all partitions $(\mathscr{B}(E), \sqsubseteq)$ of some finite set E. The order $\sqsubseteq$ is *finer-than* order relation between partitions. That is for $P_1, P_2 \in \mathscr{B}(E)$ two partitions, $P_1 \sqsubseteq P_2$ if and only $(\forall E_1 \in P_1\, \exists E_2 \in P_2)\, E_1 \subseteq E_2$.

Before finishing this section, let us highlight some key differences between arbitrary pattern setups and pattern structures:

- The set of definable sets $(\mathbb{P}_{ext}, \subseteq)$ is not necessarily closed under intersection in an arbitrary pattern setup as shown in Fig. 3.7.
- In a pattern setup, not all subsets of objects are coverable as shown in Example 3.15. However, in a pattern structure, all subsets $A \subseteq \mathcal{G}$ are coverables since $cov(A) \supseteq cov(\mathcal{G}) \neq \emptyset$. In fact, $\mathcal{G}$ is always definable in a pattern structure since $ext\left(\bigsqcap \delta[\mathcal{G}]\right) = \mathcal{G}$.
- The most important difference is the fact that, by definition, the greatest common description covering a set of objects does not necessarily exist in an arbitrary pattern setup. Generally, the notion of greatest common descriptions is relaxed to the notion of support-closed patterns [29] in arbitrary pattern setups (e.g. closed sequences in [166]). The following section has for aim to study support-closed patterns in arbitrary pattern setups.

[4] $\mathfrak{C}(\mathbb{R}^m)$ is the set of all convex subsetes of $\mathbb{R}^m$.

3.5 Support-closedness in Pattern Setups

Let $\mathbb{P} = (\mathcal{G}, \underline{\mathcal{D}}, \delta)$ be a pattern setup. Two descriptions $c, d \in \mathcal{D}$ may be equivalent, i.e. $ext(c) = ext(d)$. If $c \sqsubseteq d$ we will say that d is more informative than c on the instances that they cover. Hence, d may be preferred than c for some descriptive task. Conversely, c may also be preferred than d if someone need more generic descriptions. Definition 3.22 presents two kinds of descriptions.

Definition 3.22. A description $d \in \mathcal{D}$ is said to be:

- a **maximal generator** or **(support-)closed**[5] iff any more restrictive description $d \sqsubsetneq c$ covers strictly less instances than d, i.e.:

$$(\forall c \in \mathcal{D})\ d \sqsubsetneq c \Longrightarrow ext(c) \subsetneq ext(d).$$

- a **minimal generator** iff for any less restrictive description $c \sqsubsetneq d$ covers strictly more instances than d, i.e.:

$$(\forall c \in \mathcal{D})\ c \sqsubsetneq d \Longrightarrow ext(c) \subsetneq ext(d).$$

Example 3.21. Consider the pattern setup depicted in Fig. 3.7. The description $d =$ "bb'' is support-closed. Indeed, we have $ext(d) = \{g_2, g_4\}$ and any more restrictive description $d \sqsubsetneq c$ covers strictly less instances than d (i.e. check the extents of "bbb'', "bbc'', "cbb'', "abb'', "bba''). Description $d =$ "bb'' is also a minimal generator since $ext(\text{“}b'') = \{g_1, g_2, g_4\} \supsetneq ext(\text{“}bb'')$.

We have $ext(\text{“}cb'') = ext(\text{“}cbb'') = ext(\text{“}cbba'') = \{g_2\}$. Hence "$cbb''$ is neither support-closed not a minimal generator. Clearly "cb'' is a minimal generator but not support-closed ($ext(\text{“}c'') = ext(\text{“}b'') = \{g_1, g_2, g_4\}$). Conversely, $ext(\text{“}cbba'')$ is support-closed but not a minimal generator. □

We will now investigate further the notion of support-closed patterns and will show that this notion is tightly linked with the notion of **maximal covering descriptions** in pattern setups or more particularly closed descriptions in pattern structures.

3.5.1 On Maximal Covering Descriptions

In pattern structures, support-closed patterns coincide exactly with closed descriptions (i.e. fixpoints of $int \circ ext$) since int takes a subset of objects to the greatest common description. Hence, any more restrictive description will cover strictly less instances. However, when we consider an arbitrary pattern setup, such a maximum common description may not exist (see Example 3.19). One straightforward generalization is to associate to a subset of objects the set of its maximal common descriptions (see Definition 3.23).

[5] The term support-closed was introduced in [29].

Definition 3.23. The set of **maximal covering (common) descriptions** of a subset $A \subseteq \mathcal{G}$, denoted by $cov^*(A)$, is given by:

$$cov^* : A \mapsto max(cov(A)) = max\left(\delta[A]^{\ell}\right) = \text{minf}(\delta[A])$$

We show in Theorem 3.3 that support-closed patterns are maximal common descriptions.

Theorem 3.3. A description $d \in \mathcal{D}$ is *support-closed iff* $(\exists A \subseteq \mathcal{G})\ d \in cov^*(A)$. The set of all *support-closed descriptions* is denoted $\mathbb{P}_{int}$[6] and is given by:

$$\mathbb{P}_{int} = \bigcup_{A \subseteq \mathcal{G}} cov^*(A)$$

Proof. We prove the two implications:

(⇒) Let $d \in \mathcal{D}$ be a support-closed description and let $A = ext(d)$. Hence, according to proposition 3.4 we have $d \in cov(A)$. Let us show now that $d \in cov^*(A)$. Suppose that $d \notin cov^*(A)$ that is $\uparrow d \cap cov(A) \neq \{d\}$. Since $d \in cov(A)$, there is then at least $c \in cov(A)$ such that $d \sqsubsetneq c$. Thus, in one hand and according to proposition 3.3, $ext(c) \subseteq ext(d)$. And since $c \in cov(A)$, according to proposition 3.4, $ext(c) \supseteq ext(d)$. Thus $ext(c) = ext(d)$. This is contradictory with the fact that d is support-closed ($\exists c \in \mathcal{D}$ s.t. $d \sqsubsetneq c$ and $ext(c) = ext(d)$). Therefore, $(\exists A \subseteq \mathcal{G})\ d \in cov^*(A)$.

(⇐) Suppose that $\exists A \subseteq \mathcal{G}$ s.t. $d \in max(cov(A))$. According to proposition 3.4 and since $d \in cov(A)$, we have $A \subseteq ext(d)$. Let now be $c \in \mathcal{D}$ such that $d \sqsubsetneq c$, we have $c \notin cov(A)$ since d is maximal in $cov(A)$. According to proposition 3.3 we have $ext(c) \subseteq ext(d)$. Moreover, using proposition 3.4 and since $c \notin cov(A)$ we have $A \not\subseteq ext(c)$. Since $A \subseteq ext(d)$ then $ext(c) \neq ext(d)$. Thus $\forall c \in \mathcal{D}$ such that $d \sqsubsetneq c$ we have $ext(c) \subsetneq ext(d)$; that is d is support-closed.

The formula of $\mathbb{P}_{int}$ is deduced directly. Please notice also that if there exists A s.t. $d \in cov^*(A)$ then $d \in cov^*(ext(d))$ (use (⇐) then (⇒)). Hence, $d \in \mathcal{D}$ is support-closed iff $d \in cov^*(ext(d))$. ∎

Example 3.22. Reconsider Example 3.12, we have $cov(\{g_2, g_4\}) = \{\text{“}b'', \text{“}bb'', ''c''\}$. Hence, the maximal covering ones are given by $cov^*(\{g_2, g_4\}) = \{\text{“}bb'', \text{“}c''\}$. □

Note 3.7. According to the proof, a description $d \in \mathcal{D}$ is support-closed iff $d \in cov^*(ext(d))$. □

Maximal covering descriptions are the generalization of closed patterns in a pattern setup. However, someone need to be cautious when using them. We will now investigate some eventual problem that we can have when dealing with maximal covering descriptions.

[6] Even if the *intent* operator does not exist in a pattern setup, we use the notation $\mathbb{P}_{int}$ for support-closed descriptions since they coincide with closed patterns in a pattern structure, i.e. $cov^* : \wp(\mathcal{G}) \to \wp(\mathcal{D}), A \mapsto \{int(A)\}$.

3.5.1.1 Maximal covering descriptions of an extent do not have the same extent

In a pattern structure $\mathbb{P}$, if a subset A is definable ($A \in \mathbb{P}_{ext}$) then we have $ext(int(A)) = A$. However, for arbitrary pattern setup $\mathbb{P}$, maximal covering descriptions for $A \in \mathbb{P}_{ext}$ could have different extents as the following example demonstrates.

Example 3.23. Reconsider Example 3.12, we have $\{g_2, g_4\} \in \mathbb{P}_{ext}$ and $cov^*(\{g_2, g_4\}) = \{\text{“}bb''\text{,“}c''\}$. However, we have $ext(\text{“}bb'') = \{g_2, g_4\} \neq ext(\text{“}c'') = \{g_1, g_2, g_4\}$. In fact, since “$c''$ is support-closed, we have “$c'' \in cov^*(\{g_1, g_2, g_4\})$ (see Note 3.7) □

The example above shows also that a support-closed pattern could appear in the set of maximal covering descriptions of more than one definable set [45].

3.5.1.2 Support-closed descriptions may induce the same extent

In pattern structures, there is a one-to-one correspondence between intents in $\mathbb{P}_{int}$ and extents in $\mathbb{P}_{ext}$. In pattern setups, two distinct support-closed patterns in $\mathbb{P}_{int}$ may induce the same extent as shown in the following Example.

Example 3.24. Reconsider Example 3.12, we have $cov^*(\{g_1, g_2, g_4\}) = \{\text{“}b''\text{,“}c''\}$. Therefore, according to Theorem 3.3, descriptions “b'' and “c'' are support-closed. It is clear that $ext(\text{“}b'') = ext(\text{“}c'') = \{g_1, g_2, g_4\}$ In other words, two support-closed descriptions may be equivalent, i.e. have the same extent.

3.5.1.3 Maximal covering descriptions do not induce all the definable sets

Often, support-closed descriptions are used as a condensed representation for all (frequent) patterns. In other words, with $\mathbb{P}_{int}$ denoting the set of all support-closed patterns, we have:

$$ext[\mathbb{P}_{int}] = \mathbb{P}_{ext} \tag{3.1}$$

While this equation holds for any pattern structure. This property does no longer hold for an arbitrary pattern setup as Example 3.25 shows.

Example 3.25. Let be the description language $(\mathcal{D}, \sqsubseteq)$ presented in Fig. 3.10 (right) and given by:

- $\mathcal{D} = \{a, b\} \cup \{c_i \mid i \in \mathbb{N}\}$,
- $(\forall i \in \mathbb{N})\ c_i \sqsubseteq c_{i+1}$, $c_i \sqsubseteq a$ and $c_i \sqsubseteq b$.

Let be the pattern setup $\mathbb{P} = (\mathcal{G}, (\mathcal{D}, \sqsubseteq), \delta)$ presented in Fig. 3.10 where $\mathcal{G} = \{g_1, g_2\}$, $\delta(g_1) = a$ and $\delta(g_2) = b$. Since $cov(\{g_1, g_2\}) = \{c_i \mid i \in \mathbb{N}\}$ is an infinitely ascending chain, there is no maximal common description covering both g_1 and g_2, i.e. $cov^*(\{g_1, g_2\}) = \emptyset$. However, it is clear that for any $i \in \mathbb{N}$, $ext(c_i) = \{g_1, g_2\}$. Hence, two objects in a pattern setup could have no maximal common descriptions even if they share common descriptions. Furthermore, it is easy to see that we have:

$$\mathbb{P}_{ext} = ext[\mathcal{D}] = \{\{g_1\}, \{g_2\}, \{g_1, g_2\}\}$$

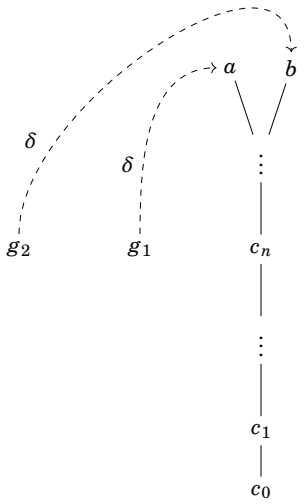


Figure 3.10: A problematic pattern setup $(\mathcal{G},(\mathcal{D},\sqsubseteq),\delta)$ with $\mathcal{G}=\{g_1,g_2\}$, $(\mathcal{D},\sqsubseteq)$ is the poset depicted right and the δ is depicted by dashed arrows (see Example 3.25).

However, since the set of support-closed patterns is given by $\mathbb{P}_{int}=\{a,b\}$ we have:

$$\mathbb{P}_{int}=\{a,b\} \;\Rightarrow\; ext[\mathbb{P}_{int}]=\{\{g_1\},\{g_2\}\}\subsetneq\mathbb{P}_{ext}$$

In other words, Equation 3.1 does no longer hold. □

3.5.2 On Upper-Approximation Extents

Going back to pattern structures, the closure operator $ext\circ int$ takes any subset $A\subseteq\mathcal{G}$ to the smallest definable set $ext(int(A))$ enclosing it as stated by Proposition 3.7. This fact is used to enumerate all definable sets via the closure operator (see Section 5.3) when considering pattern structure with finite set of objects. From Rough Set Theory [140] perspective, the set $ext(int(A))$ can be seen as the upper approximation of an arbitrary and potentially non definable set A in $\mathbb{P}_{ext}$. However, when it comes to an arbitrary pattern setup, a non-definable set A may have many *minimal* definable sets enclosing it or no one if it is non-coverable (see Example 3.26). Definition 3.24 formalizes this second generalization.

Definition 3.24. The **set of upper-approximation extents** of a subset $A\subseteq\mathcal{G}$, denoted by $\overline{A}$, is given by the set of minimal definable sets in $\mathbb{P}_{ext}$ enclosing A:

$$\overline{A}=min(\{E\in\mathbb{P}_{ext}\mid A\subseteq E\})=min(\uparrow A\cap\mathbb{P}_{ext}).$$

Example 3.26. Consider Example 3.12, the upper approximations of subset $A=\{g_2,g_4\}$ is given by $\overline{A}=\{A\}$ since A is definable. For the set $B=\{g_1,g_2\}$, we have $\overline{B}=\{\{g_1,g_2,g_3\},\{g_1,g_2,g_4\}\}$ that is B has two upper-approximation extents. For $C=\{g_3,g_4\}$, it is clear that there is no definable set in $\mathbb{P}_{ext}$ enclosing C (see Fig. 3.7 (right)), thus $\overline{C}=\emptyset$. □

It should be noticed that, according to Proposition 3.4, we have $(\forall A \subseteq \mathcal{G})\ \overline{A} = min(ext[cov(A)])$. Moreover, in a pattern structure, we have $\overline{A} = \{ext(int(A))\}$ for all $A \subseteq \mathcal{G}$.

3.5.3 Linking Upper-Approximation Extents to Support-closed Patterns

An important question to investigate now is: *"What is the relationship between upper-approximation extents and support-closed patterns?"*

We have seen before that on the one hand cov^* operator is somehow a generalization of pattern structure int *operator* in a pattern setup and on the other hand, *upper-approximation extents* operator is a generalization of pattern structure closure operator $ext \circ int$. Indeed, in a pattern structure we have for $A \subseteq \mathcal{G}$, $cov^*(A) = \{int(A)\}$ and $\overline{A} = \{ext(int(A))\}$. That is:

$$\overline{A} = ext[cov^*(A)]$$

One judicious question is that, *does this property still hold for an arbitrary pattern setup?* The answer is negative. Indeed, we have seen in Example 3.23 and Example 3.26 that for $A = \{g_2, g_4\}$, we have on the one hand $\overline{A} = \{A\}$ and on the other hand:

$$ext[cov^*(A)] = \{ext(\text{“}bb''), ext(\text{“}c'')\} = \{\{g_2, g_4\}, \{g_1, g_2, g_4\}\} \neq \overline{A}$$

The problem is even worse as shown in Example 3.25 where for $A = \{g_1, g_2\}$ we have $\overline{A} = \{\{g_1, g_2\}\}$ since A is definable. However, $cov^*(A) = \emptyset$ making $ext[cov^*(A)] = \emptyset$.

The latter observation is directly linked with the fact that in an arbitrary pattern setup we do not have the fact that any common description subsumes at least one maximal common description. Formally, the property that for any subset $A \subseteq \mathcal{G}$, we have $cov(A) =\downarrow cov^*(A)$ (i.e. $\delta[A]^\ell =\downarrow max(\delta[A]^\ell)$ does not hold in an arbitrary pattern setup but holds in a pattern structure (i.e. $int(A)$ is the maximum of the lower-ideal $cov(A)$). We will see that requiring this property is directly linked to the notion of **multilattices** presented in Section 2.3.5. The next section will introduce the notion of **pattern multistructures** that rely on multilattices.

3.6 Pattern Multistructures

In Section 3.5, we have seen that arbitrary pattern setups may have some issues regarding the maximal common descriptions and their expressiveness with regard the extents they induce (see Section 3.5.1.3). This is due to the fact that pattern setups are too permissive. On the other hand, while pattern structures provide rather strong and useful properties, they are somehow too restrictive since they cannot model some usual pattern spaces as the one of sequential patterns. We propose a new structure, dubbed **Pattern Multistructure**, that lies between *pattern setups* which rely only on arbitrary posets and *pattern structures* which rely on lattices. We will see that pattern multistructures solve many of the aforementioned issues including the ones presented in section 3.5.1.3 and section 3.5.3.

3.6.1 Basic Definitions and Properties

Definition 3.25. A pattern setup $\mathbb{P} = (\mathcal{G}, (\mathcal{D}, \sqsubseteq), \delta)$ is said to be a **pattern multistructure** if for any $A \subseteq \mathcal{G}$, $cov(A)$ is *maximal-handle* (cf. definition 2.10). In other words:

$$(\forall A \subseteq \mathcal{G})\ cov(A) = {\downarrow}\, cov^*(A)$$

Note 3.8. Note that in a pattern multistructure, we have $cov(\emptyset) = \emptyset^\ell = \mathcal{D}$ is maximal-handle. Hence, we have $\mathcal{D} = {\downarrow}\, max(\mathcal{D})$ or equivalently every chain in $(\mathcal{D}, \sqsubseteq)$ is upper-bounded. □

A pattern multistructure adds an additional condition on a pattern setup which is the following: knowing maximal common descriptions covering all elements of a set of objects A allows us to deduce using the order $\sqsubseteq$ every single covering description.

It is clear that all pattern structures are by definition pattern multistructures. Graphs ordered by subgraph isomorphism relation introduced in [110] induce a pattern multistructure on the set of graphs, but not a pattern structure (a pattern structure is induced on sets of graphs incomparable w.r.t. subgraph isomorphism). Same remark holds for sequential patterns [39, 48]. This is under the assumption of the existence of a largest element $\top$ subsumed by all sequences/graphs (see Example 3.27).

Example 3.27. Reconsider the pattern setup presented in Example 3.12. Since only finite sequences are considered in the description space, we have: $cov^*(\emptyset) = \emptyset$ even if $cov(\emptyset) = \mathcal{D}$. Thus, the considered pattern setup in Example 3.12 is *not a pattern multistructure* due to the empty set. The common trick to handle the empty set is to enrich $\mathcal{D}$ with a largest element $\top \triangleq \bigvee \mathcal{D}$ if it does not exist, i.e. apply a *dual lifting* of $\mathcal{D}$ (cf. Section 2.3.1). In such a case, we have $cov^*(\emptyset) = \{\top\}$. □

Another less common pattern language is presented below.

Example 3.28. Consider $\mathcal{D}_s := \{[a, a+l] \times [b, b+l] \mid a, b \in \mathbb{R}, l \in \mathbb{R}^+\} \cup \mathbb{R}^2$ the pattern language of closed squares in $\mathbb{R}^2$. This language can be seen as a neighborhood pattern language [86] where the used norm is the ∞-norm. Let $\mathbb{P} := \big(\{g_1, g_2\}, (\mathcal{D}_s, \supseteq), \delta\big)$ be a pattern setup where

$\delta(g_1) = [2,5] \times [1,4]$ and $\delta(g_2) = [3,6] \times [3,6]$ (i.e. the dark gray rectangles depicted in Fig. 1.4). We have, $cov^*(\{g_1, g_2\}) = \{[1+l, 6+l] \times [1,6] \mid l \in [0,1]\}$ is infinite (see Fig. 1.4). Still, for any $d \in cov(\{g_1, g_2\})$ one can always find $d^* \in cov^*(\{g_1, g_2\})$ s.t. $d \supseteq d^*$ making pattern setup $\mathbb{P}$ a pattern multistructure. □

Let us reconsider the question: *"What is the link between maximal covering descriptions and upper-approximations extents in a pattern multistructure?"*. Before answering this question, we shall start by stating the following Lemma.

Lemma 3.2. Let $(P, \leq)$ and $(Q, \leq)$ be two posets and let $f : P \to Q$ be an order-reversing mapping. We have for any $S \subseteq P$ that $\uparrow f[\downarrow S] = \uparrow f[S]$.

Proof. Recall that $\uparrow$ and $\downarrow$ are closure operators (cf. Proposition 2.7). Let us start by showing that $\uparrow f[S] \subseteq \uparrow f[\downarrow S]$. Since $S \subseteq \downarrow S$, we have $f[S] \subseteq f[\downarrow S]$. Since $\uparrow$ is monotone, we have $\uparrow f[S] \subseteq \uparrow f[\downarrow S]$. It remains to show that $\uparrow f[\downarrow S] \subseteq \uparrow f[S]$. Let $u \in \uparrow f[\downarrow S]$, that is $\exists v \in f[\downarrow S]$ s.t. $v \sqsubseteq u$. Since $v \in f[\downarrow S]$, then $\exists x \in \downarrow S$ s.t. $v = f(x)$. Hence $\exists y \in S$ s.t. $x \leq y$. Using the fact that f is an anti-embedding, we obtain that $f(y) \sqsubseteq f(x) \sqsubseteq u$. In other words, $\exists w \in f[S]$ s.t. $w \sqsubseteq u$. This is equivalent to say that $u \in \uparrow f[S]$. We conclude hence that $\uparrow f[\downarrow S] \subseteq \uparrow f[S]$. ▌

Theorem 3.4 links between maximal common descriptions and upper-approximation extents in pattern multistructures.

Theorem 3.4. For any *pattern multistructure* $\mathbb{P}$ we have: $(\forall A \subseteq \mathcal{G})\ \overline{A} = min(ext[cov^*(A)])$

Proof. The proof of the theorem is a straightforward application of Lemma 2.1 and Lemma 3.2. Let $A \subseteq \mathcal{G}$, since $\mathbb{P}$ is a pattern multistructure, then:

$$cov(A) = \downarrow max(cov(A)) \quad \Rightarrow ext[cov(A)] = ext[\downarrow max(cov(A))]$$
$$\Rightarrow \uparrow ext[cov(A)] = \uparrow ext[\downarrow max(cov(A))]$$

Since $ext : \mathcal{D} \to \wp(\mathcal{G})$ is an order reversing, then using Lemma 3.2 we have:

$$\uparrow ext[\downarrow max(cov(A))] = \uparrow ext[max(cov(A))] \Rightarrow \uparrow ext[cov(A)] = \uparrow ext[max(cov(A)) \Rightarrow$$
$$min(\uparrow ext[cov(A)]) = min(\uparrow ext[max(cov(A)])$$

Using Lemma 2.1 we obtain $\overline{A} = min(ext[cov(A)]) = min(ext[max(cov(A))))$. ▌

Another important observation related to Example 3.25 is the fact that the support-closed patterns in a pattern setup does not hold all the information about the definable sets. Theorem 3.5 states that this is no longer the case for pattern multistructures.

Theorem 3.5. Given a pattern multistructure $\mathbb{P}$ for which the set of support-closed patterns is $\mathbb{P}_{int}$ (cf. Theorem 3.3), we have: $\mathbb{P}_{ext} = ext[\mathbb{P}_{int}]$.

Proof. Recall that $\mathbb{P}_{int} = \bigcup_{B \subseteq \mathcal{G}} ext[cov^*(B)]$. Since $\mathbb{P}_{int} \subseteq \mathcal{D}$ and by definition $\mathbb{P}_{ext} = ext[\mathcal{D}]$. It is clear that $ext[\mathbb{P}_{int}] \subseteq \mathbb{P}_{ext}$. It remains to show that $\mathbb{P}_{ext} \subseteq ext[\mathbb{P}_{int}]$. Let $A \in \mathbb{P}_{ext}$, then $\exists d \in cov(A)$ s.t. $A = ext(d)$. Since $\mathbb{P}$ is a pattern multistructure, we have $cov(A) =\downarrow cov^*(A)$. Then, we have a support-closed pattern $d^* \in cov^*(A) \subseteq \mathbb{P}_{int}$ s.t. $d \sqsubseteq d^*$. Hence, $ext(d^*) \subseteq ext(d)$. Moreover, since $cov^*(A) \subseteq cov(A)$, we have $d^* \in cov(A)$. Therefore, $A = ext(d) \subseteq ext(d^*)$. We obtain thus $A = ext(d) = ext(d^*)$, that is $A \in ext[\mathbb{P}_{int}]$. ∎

Similarly to Theorem 3.2 for pattern structures, Theorem 3.6 connects *multilattices* (see Definition 2.18) with *pattern multistructures*. It states that (complete) meet-multisemilattices are to pattern multistructures what (complete) lattices are to pattern structures.

Theorem 3.6. Let $\underline{\mathcal{D}} = (\mathcal{D}, \sqsubseteq)$ be a poset, the following properties are equivalent:

- For any finite set $\mathcal{G} \neq \emptyset$ and any $\delta \in \mathcal{D}^{\mathcal{G}}$, $(\mathcal{G}, \underline{\mathcal{D}}, \delta)$ is a pattern multistructure.
- $\underline{\mathcal{D}}$ is a meet-multisemilattice having all its maximal elements (i.e. $\mathcal{D} =\downarrow max(\mathcal{D})$)

More generally, the following properties are equivalent:

- For any set $\mathcal{G} \neq \emptyset$ and any $\delta \in \mathcal{D}^{\mathcal{G}}$, $(\mathcal{G}, \underline{\mathcal{D}}, \delta)$ is a pattern multistructure.
- $\underline{\mathcal{D}}$ is a complete meet-multisemilattice.

Proof. Recall that $\mathbb{P} = (\mathcal{G}, \underline{\mathcal{D}}, \delta)$ is a pattern multistructure *iff* for any subset $S \subseteq \delta[\mathcal{G}]$, S^ℓ is maximal-handle. The proof of this theorem follows the same spirit of Theorem 3.2's proof where the existence of the meet (i.e. S^ℓ is maximum-handle) is replaced by S^ℓ is maximal-handle. ∎

Note 3.9. Pattern multistructures are named so since they rely on meet-multisemilattices. □

3.6.2 Some Issues with Pattern Multistructures

We have seen earlier in this section that some issues related to pattern setups no longer exists for pattern multistructures. However, it should be noticed that the two issues presented in section 3.5.1.1 and section 3.5.1.2 still exist as the examples used rely on the pattern setup presented example 3.12 which induce a pattern multistructure (see example 3.27). We will investigate now two additional issues that pattern multistructures may have.

3.6.2.1 Maximal covering descriptions of an extent could be infinite

The following Example shows that even with pattern multistructures with finite set of objects, upper-approximations extents are not *computable* using Theorem 3.4 as the set of maximal covering descriptions may be infinite.

Example 3.29. Consider the description space $(\mathcal{D}, \sqsubseteq)$ depicted in Fig. 3.11 and let be the pattern multistructure $(\mathcal{G}, (\mathcal{D}, \sqsubseteq), \delta)$ where $\mathcal{G} = \{g_1, g_2\}$, $\delta(g_1) = a_\alpha$ and $\delta(g_2) = a_\beta$. We have $cov(\{g_1, g_2\}) = \{b_i \mid i \in \mathbb{N}\}$ and since $cov(\{g_1, g_2\})$ is an antichain, we conclude that $cov^*(\{g_1, g_2\}) = \{b_i \mid i \in \mathbb{N}\}$. □

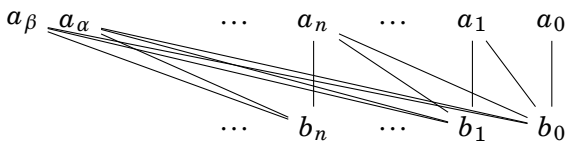


Figure 3.11: A complete multilattice with an infinite antichain. We have $(\forall i,j \in \mathbb{N})\ i \leq j \Leftrightarrow b_i \sqsubseteq a_j$ and $(\forall i \in \mathbb{N})\ b_i \sqsubseteq a_\alpha$ and $b_i \sqsubseteq a_\beta$

3.6.2.2 Definable sets do not form a join-multisemilattice

Last but not least, we have seen in Section 3.4 that in the case of a pattern structure, $(\mathbb{P}_{ext}, \subseteq)$ is a complete lattice since it is closed under arbitrary intersections. One can say that the property of having the infimum in the description space has been transferred to the poset of definable sets thanks to extent operator. When it comes to a pattern setup on finite set of objects, it is clear that $(\mathbb{P}_{ext}, \subseteq)$ is a complete multilattice since it is finite. However, *does this property still holds for the case of infinite set of objects?* Unfortunately, the answer is negative as stated in Proposition 3.9. This proposition tells also that not all definable sets above A in a pattern multistructure are above at least one upper-approximation of A.

Proposition 3.9. There exists a pattern multistructure $\mathbb{P} = (\mathcal{G}, (\mathcal{D}, \sqsubseteq), \delta)$ such that $(\mathbb{P}_{ext}, \subseteq)$ is not a join-multisemilattice and in which $(\exists A \subseteq \mathcal{G})\ \uparrow \overline{A} \cap \mathbb{P}_{ext} \neq \uparrow A \cap \mathbb{P}_{ext}$.

Proof. Consider the pattern setup $\mathbb{P} = (\mathcal{G}, (\mathcal{D}, \sqsubseteq), \delta)$ where $(\mathcal{D}, \sqsubseteq)$ is the complete multilattice depicted in Fig. 3.11. Since $(\mathcal{D}, \sqsubseteq)$ is a complete multilattice (i.e. it is chain-finite), then $\mathbb{P}$ is a pattern multistructure. Consider now an infinite set $\mathcal{G} = \{g_i \mid i \in \mathbb{N}\} \cup \{g_\alpha, g_\beta\}$. The mapping δ is given by: $\delta(g_\alpha) = a_\alpha$, $\delta(g_\beta) = a_\beta$ and $(\forall i \in \mathbb{N})\ \delta(g_i) = a_i$. To show that the poset $(\mathbb{P}_{ext}, \subseteq)$ is not a join-multisemilattice we need to consider two definable sets in $\mathbb{P}_{ext}$ and show that the set of their common upper-bounds in $\mathbb{P}_{ext}$ does not have all its minimal elements. Let us compute ext for every $d \in \mathcal{D}$:

- $ext(a_\alpha) = \{g_\alpha\}$ and $ext(a_\beta) = \{g_\beta\}$.
- $(\forall i \in \mathbb{N})\ ext(a_i) = \{g_i\}$ and $(\forall i \in \mathbb{N})\ ext(b_i) = \{g_\alpha, g_\beta\} \cup \{g_j \mid j \geq i\}$.

Consider now the set of definable sets $\{\{g_\alpha\}, \{g_\beta\}\}$, it is clear that the set of their common upper-bounds (in $\mathbb{P}_{ext}$) is given by: $\{\{g_\alpha\}, \{g_\beta\}\}^u = \big\{\big\{g_\alpha, g_\beta\big\} \cup \big\{g_j \mid j \geq i\big\} \mid i \in \mathbb{N}\big\}$.

The set of upper bounds is hence an infinitely descending chain and hence does not have a minimal element, in other words: $min(\{\{g_\alpha\}, \{g_\beta\}\}^u) = \emptyset$. Hence, $(\mathbb{P}_{ext}, \subseteq)$ is not a join-multisemilattice. The proof of the second part of the proposition is straightforward. Indeed, consider the non-definable set $A = \{g_\alpha, g_\beta\}$. We do have: $\uparrow A \cap \mathbb{P}_{ext} = ext[cov(A)] = \big\{\big\{g_\alpha, g_\beta\big\} \cup \big\{g_j \mid j \geq i\big\} \mid i \in \mathbb{N}\big\}$. Hence, $\overline{A} = min(\uparrow A \cap \mathbb{P}_{ext}) = \emptyset$. That is $\uparrow A \cap \mathbb{P}_{ext} \neq \uparrow \overline{A} \cap \mathbb{P}_{ext}$. ∎

3.7 New Pattern Setups from Old Ones

In this section, we will explore three usual ways used in the literature to build new pattern setups from other. The first investigated transformation, called **completion**, allows to build pattern structures starting from pattern setups (Section 3.7.1). The next discussed transformation, called **projection**, allows to build simpler pattern setups starting from a more complex one (Section 3.7.2). Finally, we present the notion of combining pattern setups to build a more complex one thanks to the direct product of posets (Section 3.7.3).

3.7.1 Pattern Setup Completions

Example 3.12 presents a pattern setup which is not a pattern structure. However, in FCA and pattern structure literature, it is recurrent to talk about sequential pattern structures [39, 48]. In fact, instead of sequences, sets of incomparable sequences are considered which induce a richer description space. Same trick has been even used in the first paper introducing pattern structures [78] concerning graph description space ordered by subgraph isomorphism. Transforming pattern setups to pattern structures allow in general to use results from pattern structures to handle other pattern languages as for instance enumeration algorithms.

From the order-theoretic point of view, techniques *embedding* a poset into a complete lattice are called **completions** (see Definition 2.26). Different natural completions exist in the literature. For instance, the Dedekind-MacNeille completion [53] takes an arbitrary poset to the smallest complete lattice containing it. The usual trick used in FCA and Pattern Structure literature to augment a base pattern setup to a pattern structure is tightly linked to the antichain embeding presented in Section 2.4.4.3.

3.7.1.1 Pattern setup completion by antichain embedding

We define below the most common trick used in the FCA literature which can be called *pattern setup antichain embedding*.

Definition 3.26. Let $\mathbb{P} = (\mathcal{G}, \underline{\mathcal{D}}, \delta)$ be a *pattern setup*, the **antichain embedding** of $\mathbb{P}$ is the *pattern setup* denoted by $\mathbb{P}^{\triangledown}$ and given by:

$$\mathbb{P}^{\triangledown} = \left(\mathcal{G}, (\mathscr{A}(\mathcal{D}), \leqq), \delta^{\triangledown} : g \mapsto \{\delta(g)\}\right)$$

where $(\mathscr{A}(\mathcal{D}), \leqq)$ is given in Definition 2.29.

In Section 2.4.4.3, we have mentioned that $(\mathscr{A}(\mathcal{D}), \leqq)$ is a lattice when $(\mathcal{D}, \sqsubseteq)$ is a finite poset (i.e. a sufficient condition). However, given an arbitrary pattern setup on an infinite description space, $\mathbb{P}^{\triangledown}$ is not always guaranteed to be a pattern structure. We show in Theorem 3.7 a necessary and sufficient condition on $\mathbb{P}$ that makes $\mathbb{P}^{\triangledown}$ a *pattern structure*. Here $ext^{\triangledown}$ and $int^{\triangledown}$ denote extent and intent of $\mathbb{P}^{\triangledown}$, respectively.

Theorem 3.7. Let $\mathbb{P} = (\mathcal{G}, \underline{\mathcal{D}}, \delta)$ be a pattern setup. The *antichain embedding* of $\mathbb{P}$ is a *pattern structure*, i.e. a **completion**, *if and only if* $\mathbb{P}$ is a *pattern multistructure*. In this case:

$$(\forall S \in \mathscr{A}(\mathcal{D}))\, ext^{\triangledown}(S) = \bigcap ext[S] \qquad (\forall A \subseteq \mathcal{G})\, int^{\triangledown}(A) = \text{minf}(\delta[A]) = cov^{*}(A)$$

and we have $\mathbb{P}^{\triangledown}_{ext} = \{\bigcap S \mid S \subseteq \mathbb{P}_{ext}\}$ and $\bigcap \emptyset = \mathcal{G} \in \mathbb{P}^{\triangledown}_{ext}$.

Proof. Let us show that: $\mathbb{P}$ is a pattern multistructure $\Leftrightarrow$ $\mathbb{P}^{\triangledown}$ is a pattern structure.

Recall that $\mathbb{P}^{\triangledown}$ is a pattern structure **iff** every subset of $\delta^{\triangledown}[\mathcal{G}]$ has a meet in $(\mathscr{A}(\mathcal{D}), \leq)$. For $A \subseteq \mathcal{G}$ we have: $\delta^{\triangledown}[A]^{\ell} = \{S \in \mathscr{A}(\mathcal{D}) \mid (\forall g \in A) S \subseteq\!\downarrow \delta(g)\} = \{S \in \mathscr{A}(\mathcal{D}) \mid S \subseteq \delta[A]^{\ell}\}$. where $\delta[A]^{\ell}$ and $\delta^{\triangledown}[A]^{\ell}$ denote respectively the lower bounds of $\delta[A]$ w.r.t. $\sqsubseteq$ and the lower bounds of of $\delta^{\triangledown}[A]$ w.r.t. $\leq$ (recall that $\delta[A]^{\ell} = \bigcap_{g \in A} \downarrow \delta(g)$). In this proof $\downarrow$ refers to the down-closure related to $\sqsubseteq$.

We show each implication independently:

- ($\Rightarrow$) Let $A \subseteq \mathcal{G} : \delta[A]^{\ell} =\!\downarrow max(\delta[A]^{\ell})$. Thus $\delta^{\triangledown}[A]^{\ell} = \{S \in \mathscr{A}(\mathcal{D}) \mid S \subseteq\!\downarrow max(\delta[A]^{\ell})\} = \{S \in \mathscr{A}(\mathcal{D}) \mid S \leq max(\delta[A]^{\ell})\}$. Since $max(\delta[A]^{\ell}) \in \mathscr{A}(\mathcal{D})$, so $max(\delta[A]^{\ell})$ is the meet of $\delta^{\triangledown}[A]$ in $\mathscr{A}(\mathcal{D})$.
- ($\Leftarrow$) $\mathbb{P}^{\triangledown}$ is a pattern structure is **equivalent** to say: $\forall A \subseteq \mathcal{G}$, $\delta^{\triangledown}[A]$ has a meet $M \in \mathscr{A}(\mathcal{D})$. That is, $\exists M \in \delta^{\triangledown}[A]^{\ell}$ for $A \subseteq \mathcal{G}$: $\forall S \in \mathscr{A}(\mathcal{D}) : S \subseteq \delta[A]^{\ell} \Leftrightarrow S \subseteq\!\downarrow M$. Particularly, for $S = \{d\}$ with $d \in \mathcal{D}$, we deduce that $\forall d \in \delta[A]^{\ell} : d \in\!\downarrow M$. Thus, $\delta[A]^{\ell} \subseteq\!\downarrow M$. Moreover, since $M \subseteq \delta[A]^{\ell}$ ($M \in \delta^{\triangledown}[A]^{\ell}$) and $\downarrow$ is a closure operator on $(\wp(\mathcal{D}), \subseteq)$ we have by monotony $\downarrow M \subseteq \delta[A]^{\ell} \subseteq\!\downarrow M$ (note that $\downarrow \delta[A]^{\ell} = \delta[A]^{\ell}$). We conclude that we have $\delta[A]^{\ell} =\!\downarrow M$. Using Lemma 2.3 we obtain $\delta[A]^{\ell} =\!\downarrow max(\delta[A]^{\ell})$.

We conclude the equivalence.

Let us now determine $int^{\triangledown}$ and $ext^{\triangledown}$. The previous proof has shown that for $A \subseteq \mathcal{G}$ the meet of $\delta^{\triangledown}[A]$ is $max(\delta[A])^{\ell}$. i.e.:

$$int^{\triangledown}(A) = max(\delta[A]^{\ell}) = cov^{*}(A)$$

For $ext^{\triangledown}$ operator, let $S \in \mathscr{A}(\mathcal{D})$. We have:

$$\begin{aligned} ext^{\triangledown}(S) &= \{g \in \mathcal{G} \mid S \leq \sigma(g)\} = \{g \in \mathcal{G} \mid S \subseteq\!\downarrow \delta(g)\} \\ &= \{g \in \mathcal{G} \mid (\forall d \in S) d \sqsubseteq \delta(g)\} = \bigcap_{d \in S} ext(d) = \bigcap ext[S] \end{aligned}$$

Let us show that $\mathbb{P}^{\triangledown}_{ext} = \{\bigcap S \mid S \subseteq \mathbb{P}_{ext}\}$. The property $\mathbb{P}^{\triangledown}_{ext} \subseteq \{\bigcap S \mid S \subseteq \mathbb{P}_{ext}\}$ holds by definition of $ext^{\triangledown}$. For the inverse inclusion, it is sufficient to show that $\mathbb{P}_{ext} \subseteq \mathbb{P}^{\triangledown}_{ext}$ (since $(\mathbb{P}^{\triangledown}_{ext}, \subseteq)$ is closed under intersection). Let $A \in \mathbb{P}_{ext}$. $\exists d \in \mathcal{D}$ s.t. $A = ext(d)$. Since $\{d\} \in \mathscr{A}(\mathcal{D})$, and $ext^{\triangledown}(\{d\}) = ext(d) = A$. We conclude that $A \in \mathbb{P}^{\triangledown}_{ext}$. Hence, $\mathbb{P}^{\triangledown}_{ext} = \{\bigcap S \mid S \subseteq \mathbb{P}_{ext}\}$.

3.7.1.2 Pattern setup direct completion

We show here that there is a completion that transforms any pattern setup to a pattern structure without demanding any additional property.

Theorem 3.8. The **direct completion** of $\mathbb{P} = (\mathcal{G}, \underline{\mathcal{D}}, \delta)$ is the *pattern structure*:

$$\mathbb{P}^{\blacktriangledown} \quad = \quad (\mathcal{G}, (\wp(\mathcal{D}), \subseteq), \delta^{\blacktriangledown} : g \mapsto \downarrow \delta(g))$$

where $(\forall S \subseteq \mathcal{D})\, ext^{\blacktriangledown}(S) = \bigcap ext[S]$ and $(\forall A \subseteq \mathcal{G})\, int^{\blacktriangledown}(A) = cov(A) = \delta[A]^{\ell}$. The set of definable sets is given by $\mathbb{P}^{\blacktriangledown}_{ext} = \{\bigcap S \mid S \subseteq \mathbb{P}_{ext}\}$.

Proof. Let us show that the pattern setup $\mathbb{P}^{\blacktriangledown}$ is a pattern structure. Let $A \subseteq \mathcal{G}$. We need to show that $\delta^{\triangledown}[A]$ has a meet in $(\wp(\mathcal{D}), \subseteq)$. We have:

$$\delta^{\triangledown}[A]^{\ell} = \{S \subseteq \mathcal{D} \mid (\forall g \in A) S \subseteq \downarrow \delta(A)\} = \{S \subseteq \mathcal{D} \mid S \subseteq \delta[A]^{\ell}\}$$

Since $\delta[A]^{\ell} \in \delta^{\triangledown}[A]^{\ell}$, we conclude $\delta[A]^{\ell}$ is the meet of $\delta^{\triangledown}[A]^{\ell}$, Hence:

$$int^{\blacktriangledown}(A) = \delta[A]^{\ell} = cov(A)$$

For the extent operator $ext^{\blacktriangledown}$, let $S \in \wp(\mathcal{D})$. We have

$$ext^{\blacktriangledown}(S) = \{g \in \mathcal{G} \mid S \subseteq \downarrow \delta(g)\} = \{g \in \mathcal{G} \mid (\forall d \in S)\, d \sqsubseteq \delta(g)\} = \bigcap ext[S]$$

Let us show that $\mathbb{P}^{\blacktriangledown}_{ext} = \{\bigcap S \mid S \subseteq \mathbb{P}_{ext}\}$. Thanks to the definition of $ext^{\blacktriangledown}$, property $\mathbb{P}^{\triangledown}_{ext} \subseteq \{\bigcap S \mid S \subseteq \mathbb{P}_{ext}\}$ holds. For the inverse inclusion, it is sufficient to show that $\mathbb{P}_{ext} \subseteq \mathbb{P}^{\blacktriangledown}_{ext}$ (since $(\mathbb{P}^{\blacktriangledown}_{ext}, \subseteq)$ is closed under intersection). Let $A \in \mathbb{P}_{ext}$, $\exists d \in \mathcal{D}$ s.t. $A = ext(d)$. We have $ext^{\blacktriangledown}(\{d\}) = \{g \in \mathcal{G} \mid \{d\} \subseteq \downarrow \delta(g)\} = \{g \in \mathcal{G} \mid d \sqsubseteq \delta(g)\} = ext(d) = A$. We conclude that $A \in \mathbb{P}^{\blacktriangledown}_{ext}$ and $\mathbb{P}^{\blacktriangledown}_{ext} = \{\bigcap S \mid S \subseteq \mathbb{P}_{ext}\}$. ∎

Example 3.30. Fig. 3.12 depicts the concept lattice $\underline{\mathfrak{B}}(\mathbb{P}^{\triangledown})$ of the *antichain completion* of the pattern multistructure $\mathbb{P}$ considered in Fig. 3.7 (*i.e.,* the description space is augmented with the top element $\top$). For any concept (A, B), descriptions $d \in B$ in **bold** are those for which $ext(d) = A$. Please notice that while description "c" has for extent $\{g_1, g_2, g_4\}$, description "c" does belong to the concept related to the extent $\{g_2, g_4\}$. One should also notice that the underlined concepts represent those related to the non definable sets in $\mathbb{P}$ but definable in $\mathbb{P}^{\triangledown}$, i.e. $\{g_1, g_2\}$ and $\{g_1, g_2, g_3, g_4\}$ in $\mathbb{P}^{\triangledown}_{ext} \setminus \mathbb{P}_{ext}$. For instance, consider the intent of $\{g_1, g_2\}$ in the completion, each pattern d has extent $ext(d) \supsetneq \{g_1, g_2\}$. Extent $\{g_1, g_2, g_3, g_4\}$ is *non-coverable* in $\mathbb{P}$ and thus $int^{\triangledown}(\{g_1, g_2, g_3, g_4\}) = max(cov(\{g_1, g_2, g_3, g_4\})) = max(\emptyset) = \emptyset$. □

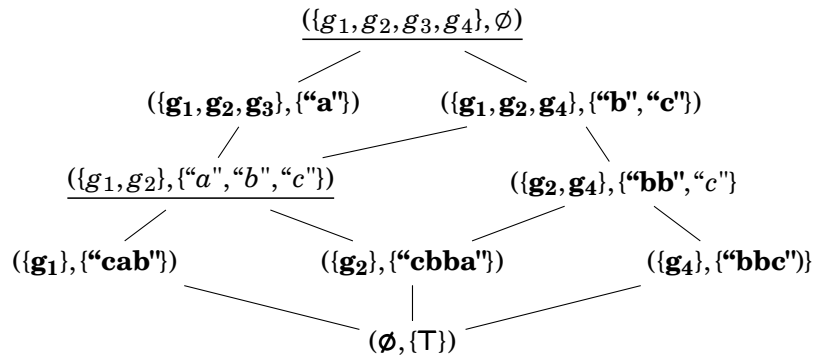


Figure 3.12: Concept lattice $\underline{\mathfrak{B}}(\mathbb{P}^{\triangledown})$.

Note that while in Example 3.30, the size difference between the set of definable sets in the base pattern setup $\mathbb{P}_{ext}$ and the set of definable sets in the antichain completion $\mathbb{P}^{\triangledown}_{ext}$ is not large (i.e. $|\mathbb{P}^{\triangledown}_{ext} \backslash \mathbb{P}_{ext}| = 2$). In some cases, the size of $\mathbb{P}^{\triangledown}_{ext}$ can be exponentially larger than $\mathbb{P}_{ext}$. Consider, for instance, the following example.

Example 3.31. Let $n \in \mathbb{N}$ with $n \geq 3$. We denote by $[n]$ the subset $[n] = \{1, 2, ..., n\}$. Let be the pattern setup $\mathbb{P} = (\mathcal{G}, (\mathcal{D}, \subseteq), \delta)$ with $\mathcal{G} = \{g_i\}_{i \in [n]}$,

$$\mathcal{D} = \{\{i\} \mid i \in [n]\} \cup \{[n] \backslash \{i\} \mid i \in [n]\}$$

and the mapping $\delta : g_i \mapsto [n] \backslash \{i\}$ for all $i \in [n]$. One can verify that we have $\mathbb{P}_{ext} = \{\{g_i\} \mid i \in [n]\} \cup \{\mathcal{G} \backslash \{g_i\} \mid i \in [n]\}$. Indeed, we have:

- $(\forall i \in [n])\ ext([n] \backslash \{i\}) = \{g_j \mid [n] \backslash \{j\} \subseteq \delta(g_j)\} = \{g_j \mid [n] \backslash \{j\} \subseteq [n] \backslash \{i\}\} = \{g_i\}$.
- $(\forall i \in [n])\ ext(\{i\}) = \{g_j \mid \{i\} \subseteq \delta(g_j)\} = \{g_j \mid \{i\} \subseteq [n] \backslash \{j\}\} = \mathcal{G} \backslash \{g_i\}$.

Hence, we have $|\mathbb{P}_{ext}| = 2n$. However, according to Theorem 3.7, we have

$$\mathbb{P}^{\triangledown}_{ext} = \left\{\bigcap S \mid S \subseteq \mathbb{P}_{ext}\right\} = \wp(\mathcal{G})$$

since $\mathbb{P}_{ext}$ contains all the coatoms of $\wp(\mathcal{G})$ (i.e. $(\forall g \in \mathcal{G})\ \mathcal{G} \backslash \{g\} \in \mathbb{P}_{ext}$) and the powerset lattice is coatomistic. It follows that $|\mathbb{P}^{\triangledown}_{ext}| = 2^n$. In other words, $\mathbb{P}^{\triangledown}_{ext}$ is exponentially larger than $\mathbb{P}_{ext}$. One should notice that the new description space associated to $\mathbb{P}^{\triangledown}$ is (order-)isomorphic to $(\wp([n]), \subseteq)$. □

3.7.2 Pattern Setup Projections

Projections has been proposed in the seminal paper on pattern structure [78]. They have for aim to simplify the pattern structure someone deals with, i.e. create a less expressive description language. They have been then revisited in [41]. While this notion has been defined for pattern structures, we extend here this notion in the more general framework of pattern setups following [41]. Please note that what we call here projections were called **o-projections** [41].

Definition 3.27. Let $\mathbb{P} = (\mathcal{G}, (\mathcal{D}, \sqsubseteq), \delta)$ be a pattern setup and ψ be a kernel operator on $(\mathcal{D}, \sqsubseteq)$ (cf. Definition 2.24). The kernel operator is then said to be a **projection on** $\mathbb{P}$ and the resulting **projected pattern setup** is denoted $\psi(\mathbb{P})$ and given by:

$$\psi(\mathbb{P}) := (\mathcal{G}, (\psi[\mathcal{D}], \sqsubseteq), \psi \circ \delta)$$

The associated operators of $\wp(\mathbb{P})$ will be indexed by ψ, e.g. ext_ψ, cov_ψ, cov^*_ψ.

Note 3.10. Since ψ is a kernel operator, recall that we have from Definition 2.24:

$$(\forall c \in \mathcal{D}\ \forall d \in \psi[\mathcal{D}])\ d \sqsubseteq c \Leftrightarrow d \sqsubseteq \psi(c) \tag{3.2}$$

□

We will see now that projected pattern setups have a tight relationship with their parent pattern structures. Let us start by the following proposition.

Proposition 3.10. A description $d \in \psi[\mathcal{D}]$ **holds** for an object $g \in \mathcal{G}$ in $\psi(\mathbb{P})$ iff it does hold for g in $\mathbb{P}$. Formally:

$$(\forall d \in \psi[\mathcal{D}]\ \forall g \in \mathcal{G})\ d \sqsubseteq \psi \circ \delta(g) \Longleftrightarrow d \sqsubseteq \delta(g)$$

Proof. The proof is a straightforward application of Equation 3.2 where $c \in \delta[\mathcal{G}]$. ▌

Many other properties stem from this basic property. Let us start by investigating the expressions of the different operators associated to the projected pattern setup.

Proposition 3.11. We have:

$$ext_\psi : \psi[\mathcal{D}] \to \wp(\mathcal{G}),\ d \mapsto ext(d) \qquad cov_\psi : \wp(\mathcal{G}) \to \wp(\psi[\mathcal{D}]),\ A \mapsto \psi[cov(A)]$$

Proof. By definition, we have: $ext_\psi : \psi[\mathcal{D}] \to \wp(\mathcal{G}), d \mapsto \{g \in \mathcal{G} \mid d \sqsubseteq \psi(\delta(g))\}$. Using Proposition 3.10, we conclude that $ext_\psi(d) = ext(d)$ for all $d \in \psi[\mathcal{D}]$.

By definition, we have: $cov_\psi : \wp(\mathcal{G}) \to \wp(\psi[\mathcal{D}]), A \mapsto \{d \in \psi[\mathcal{D}] \mid (\forall g \in A)\, d \sqsubseteq \psi(\delta(g))\}$. On the other hand, we have: $\psi[cov(A)] = \{\psi(d) \mid d \in \mathcal{D} \text{ and } (\forall g \in A)\, d \sqsubseteq \delta(g)\}$. Let us show by double inclusion that $cov_\psi(A) = \psi[cov(A)]$:

- ($\subseteq$) Let $d \in cov_\psi(A)$, then $d \in \psi[D]$ with $(\forall g \in A)\, d \sqsubseteq \psi(\delta(g))$. Using Proposition 3.10, we obtain $(\forall g \in A)\, d \sqsubseteq \delta(g)$. Hence, $d \in cov(A)$, since $\psi(d) = d$ we obtain $d \in \psi[cov(A)]$.
- ($\supseteq$) Let $c \in \psi[cov(A)]$, then $\exists d \in \mathcal{D}$ s.t. $c = \psi(d)$ and $(\forall g \in A)\, d \sqsubseteq \delta(g)$. Since ψ is order-preserving we obtain $(\forall g \in A)\, c \sqsubseteq \psi(\delta(g))$. With $c \in \psi[\mathcal{D}]$ we conclude that $c \in cov_\psi(A)$.

This conclude the theorem. ▌

An important corollary of the Proposition 3.11 is given below:

Corollary 3.1. If $A \subseteq \mathcal{G}$ is an extent in $\mathbb{P}$ then it is also an extent in $\psi(\mathbb{P})$. Formally:

$$\psi(\mathbb{P})_{ext} \subseteq \mathbb{P}_{ext}$$

Proof. The proof is straightforward: $\psi(\mathbb{P})_{ext} = ext_{\psi}[\psi[\mathcal{D}]] = ext[\psi[\mathcal{D}]] \subseteq ext[\mathcal{D}] = \mathbb{P}_{ext}$. ∎

In other words, Corollary 3.1 creates a less expressive pattern setup.

3.7.2.1 Projections on pattern structures

As presented in [78], Theorem 3.9 formalizes the fact that the projections preserve pattern structures. Before presenting the theorem, we state the following Lemma.

Lemma 3.3. Let $(\mathcal{D}, \sqsubseteq)$ be a poset and ψ be a kernel operator on $(\mathcal{D}, \sqsubseteq)$. If $S \subseteq \mathcal{D}$ has a meet $\bigsqcap S$ on $\mathcal{D}$, then $\psi[S]$ has also a meet in $(\psi(\mathcal{D}), \sqsubseteq)$ which is given by:

$$\bigsqcap\nolimits_{\psi} \psi[S] = \psi[\bigsqcap S]$$

Proof. Since $\bigsqcap S$ is the meet of S in $(\mathcal{D}, \sqsubseteq)$, we have:

$$(\forall c \in \mathcal{D}) \ ((\forall s \in S)\ c \sqsubseteq s \Leftrightarrow c \sqsubseteq \bigsqcap S)$$

Using Equation 3.2 for $c \in \psi[\mathcal{D}]$ we have both equations:

$$(\forall c \in \psi[\mathcal{D}]) \left(c \sqsubseteq \bigsqcap S \Leftrightarrow c \sqsubseteq \psi[\bigsqcap S]\right)$$
$$(\forall c \in \psi[\mathcal{D}]) \left((\forall s \in S)\ c \sqsubseteq s \Leftrightarrow (\forall s \in S)\ c \sqsubseteq \psi(s)\right)$$

Using the three above equations, we obtain:

$$(\forall c \in \psi[\mathcal{D}]) \ \left((\forall s \in S)\ c \sqsubseteq \psi(s) \Leftrightarrow (\forall s \in S)\ c \sqsubseteq s \Leftrightarrow c \sqsubseteq \bigsqcap S \Leftrightarrow c \sqsubseteq \psi[\bigsqcap S]\right)$$

We have then equivalently:

$$(\forall c \in \psi[\mathcal{D}]) \ \left(\left(\forall s' \in \psi[S]\right)\ c \sqsubseteq s' \Leftrightarrow c \sqsubseteq \psi[\bigsqcap S]\right)$$

The last statement is equivalent to say that the meet of $\psi[S]$ in $(\psi[\mathcal{D}], \sqsubseteq)$ exists and is given by $\psi[\bigsqcap S]$. ∎

Theorem 3.9. Let $\mathbb{P}$ be a pattern structure and ψ a projection on $\mathbb{P}$. The projected pattern setup $\psi(\mathbb{P})$ is also a pattern structure whose intent int_{ψ} is given by:

$$int_{\psi} : \wp(\mathcal{G}) \to \psi[\mathcal{D}], A \mapsto \psi(int(A))$$

Proof. Let $A \subseteq \mathcal{G}$. since $\mathbb{P}$ is a pattern structure, then $int(A) = \sqcap\delta[A]$ exists in $(\mathcal{D}, \sqsubseteq)$. Hence, using Lemma 3.3, the meet of $\psi \circ \delta[A]$, i.e. $int_\psi(A)$ exists in $(\psi(\mathcal{D}), \sqsubseteq)$ and is given by $\psi(int(A))$. ∎

Note 3.11. Theorem 3.9 states also the same property as the one stated in Proposition 1 in [41], that is if d is an intent in $\mathbb{P}$ then $\psi(d)$ is an intent in $\psi(\mathbb{P})$. Moreover, we have:

$$\psi(\mathbb{P})_{int} \;=\; \psi[\mathbb{P}_{int}]$$

□

Further discussion about projections of pattern structures can be found in [41] as for instance the fact that not all projections are meet-preserving since they are just simple kernel operators.

3.7.2.2 Projections on pattern multistructures

We have seen earlier that projections preserve pattern structures. However, we show here that projections do not preserve pattern multistructures.

Proposition 3.12. There exists a pattern multistructure $\mathbb{P}$ and a projection ψ on $\mathbb{P}$ such that $\psi(\mathbb{P})$ is not a pattern multistructure.

Proof. Let $\mathbb{P} = (\mathcal{G}, (\mathcal{D}, \sqsubseteq), \delta)$ be a pattern setup where $(\mathcal{D}, \sqsubseteq)$ is depicted in Fig. 3.13 (**left**):

- $(\forall i \in \mathbb{N})\ c_i \sqsubseteq c_{i+1}$.
- $(\forall i \in \mathbb{N})\ c_i \sqsubseteq a_i$.
- $(\forall i \in \mathbb{N})\ a_i \sqsubseteq e_0$ and $a_i \sqsubseteq e_1$.

The set of objects $\mathcal{G}$ is given by $\mathcal{G} = \{g_0, g_1\}$ and the mapping delta is given by: $\delta(g_j) = e_j$ for $j \in \{1,2\}$. One can check that $(\mathcal{D}, \sqsubseteq)$ is a complete multilattice. Hence, According to Theorem 3.6, the pattern setup $\mathbb{P}$ is a pattern multistructure.

Let be $\psi : \mathcal{D} \to \mathcal{D}$ the mapping depicted by dashed arrows in the Fig. 3.13 **(left)** where:

- $(\forall i \in \mathbb{N})\ \psi(a_i) = c_i$.
- $(\forall i \in \mathbb{N})\ \psi(c_i) = c_i$.
- $(\forall j \in \{0,1\})\ \psi(e_j) = e_j$.

One can check that ψ is a kernel operator on $(\mathcal{D}, \sqsubseteq)$ since it is idempotent, contractive and order-preserving. Hence, ψ is a projection on $\mathbb{P}$ and $\psi(\mathbb{P}) = (\{g_0, g_1\}, (\psi[\mathcal{D}], \sqsubseteq), \delta)$. Poset $(\psi[\mathcal{D}], \sqsubseteq)$ is depicted on the right side of Fig. 3.13 while $\psi \circ \delta = \delta$ since elements of $\delta[\mathcal{G}]$ are fixpoints of ψ. It is clear that $\psi(\mathbb{P})$ is not a pattern multistructure since: $cov_\psi(\mathcal{G}) = \{c_i \mid i \in \mathbb{N}\}$ which is an infinitely ascending chain. Hence, $max(cov_\psi(\mathcal{G})) = \emptyset$. ∎

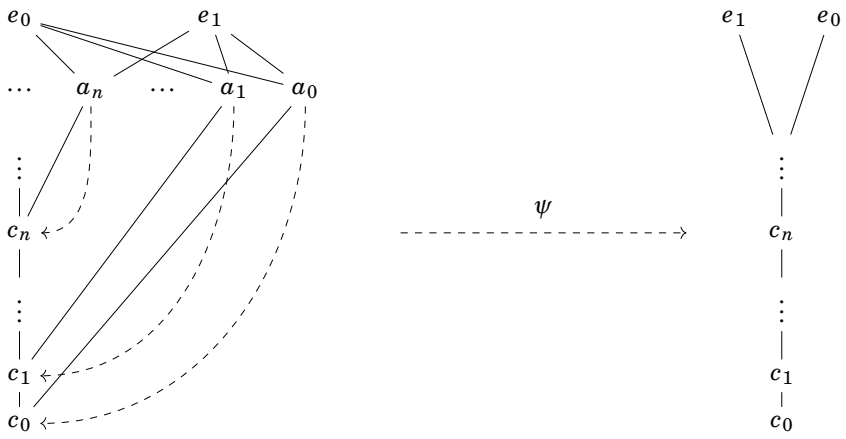


Figure 3.13: A kernel operator ψ on a complete multilattice $(\mathcal{D},\sqsubseteq)$ where dashed arrows represent the image of non fixpoints **(left)** and its image $(\psi[\mathcal{D}],\sqsubseteq)$ **(right)**.

3.7.3 Pattern Setups Direct Product

When dealing with datasets with heterogeneous attributes (e.g. categorical, numerical, boolean), we manipulate in general various pattern languages built over one or a subset of attributes. A way to combine these pattern languages is to use *"conjunctions"* of their elements. We formalize these new pattern languages built by *"conjunctions"* by the mean of pattern setups direct product defined below.

Definition 3.28. Let I be an index set and $\mathbb{P}_i = (\mathcal{G},(\mathcal{D}_i,\sqsubseteq),\delta_i)$ be pattern setups built over $\mathcal{G}$ for all $i \in I$. The **direct product** of $\{\mathbb{P}_i \mid i \in I\}$ is denoted $\times_{i\in I}\mathbb{P}_i$ and is given by:

$$\bigtimes_{i\in I}\mathbb{P}_i \;:=\; \left(\mathcal{G},\bigtimes_{i\in I}(\mathcal{D}_i,\sqsubseteq),\delta\right) \quad \textbf{with} \quad \delta : g \mapsto (\delta_i(g))_{i\in I}$$

where $\times_{i\in I}(\mathcal{D}_i,\sqsubseteq)$ is the direct product of posets $(\mathcal{D}_i,\sqsubseteq)$ (see Definition 2.20).

Proposition 3.13. We have:

$$\begin{aligned}
ext: &\quad \times_{i\in I}\mathcal{D}_i \to \wp(\mathcal{G}), \quad (d_i)_{i\in I} \mapsto \bigcap_{i\in I} ext_i(d_i)\\
cov: &\quad \wp(\mathcal{G}) \to \times_{i\in I}\mathcal{D}_i, \quad A \mapsto \bigtimes_{i\in I} cov_i(A)\\
cov^*: &\quad \wp(\mathcal{G}) \to \times_{i\in I}\mathcal{D}_i, \quad A \mapsto \bigtimes_{i\in I} cov_i^*(A)
\end{aligned}$$

Where ext_i (ext), cov_i (cov) and cov_i^* (cov^*) denote respectively the extent operator, cover operator, maximal common descriptions operator for pattern setup $\mathbb{P}_i$ ($\times_{i\in I}\mathbb{P}_i$).

Proof. We show the three expressions below:

1. ext: We have $ext((d_i)_{i\in I}) = \{g \in \mathcal{G} \mid (d_i)_{i\in I} \sqsubseteq (\delta_i(g)_{i\in I}\}$ which is equivalent by definition to $\{g \in \mathcal{G} \mid (\forall i \in I)\ d_i \sqsubseteq \delta_i(g)\}$. This last set is equal to $\bigcap_{i\in I} ext_i(d_i)$.
2. cov: We have $(d_i)_{i\in I} \in cov(A)$ is equivalent to $\forall i \in I$ and $\forall g \in \mathcal{G}$ we have $d_i \sqsubseteq \delta_i(g)$. The later is equivalent to $(\forall i \in I)\ d_i \in cov_i(A)$, i.e. $(d_i)_{i\in I} \in \bigtimes_{i\in I} cov(A)$.
3. cov^*: We have $(d_i)_{i\in I} \in cov^*(A)$ is equivalent to $\uparrow (d_i)_{i\in I} \cap cov(A) = \{(d_i)_{i\in I}\}$. The latter is equivalent to $(\forall i \in I) \uparrow d_i \cap cov_i(A) = \{d_i\}$ which is equivalent to $(\forall i \in I) d_i \in cov_i^*(A)$, i.e. $(d_i)_{i\in I} \in \bigtimes_{i\in I} cov_i^*(A)$.

This concludes the demonstration. ∎

Interestingly, analogously to the direct product of posets, the direct product of pattern setups preserve the different properties. This observation is expressed formally in the two following theorems.

Theorem 3.10. Pattern setup $\bigtimes_{i\in I} \mathbb{P}_i$ is a pattern structure if and only if for all $i \in I$, pattern setup $\mathbb{P}_i$ is a pattern structure. Moreover, with int_i (resp. int) denoting the intent operator associated to $\mathbb{P}_i$ (resp. $\bigtimes_{i\in I} \mathbb{P}_i$), we have:

$$int : \wp(\mathcal{G}) \to \bigtimes_{i\in I} \mathcal{D}_i, A \mapsto (int_i(A))_{i\in I}$$

Proof. All the pattern setups $\mathbb{P}_i$ are pattern structures is equivalent to say that for all $A \subseteq \mathcal{G}$, the meet $int_i(A) = \bigsqcap \delta_i[A]$ exists in $\mathcal{D}_i$ for all $i \in I$. This is equivalent to say:

$$(\forall A \in \mathcal{G}, \forall i \in I, \forall d_i \in \mathcal{D}_i)\ (d_i \in cov_i(A) \iff d_i \sqsubseteq int_i(A))$$

This is equivalent to:

$$\left(\forall A \in \mathcal{G}, \forall (d_i)_{i\in I} \in \mathcal{D}\right)\ \left((d_i)_{i\in I} \in cov(A) \iff (d_i)_{i\in I} \sqsubseteq (int_i(A))_{i\in A}\right)$$

Hence all pattern setups are pattern structures iff $\bigtimes_{i\in I} \mathbb{P}_i$ is a pattern structure. ∎

Theorem 3.11. Pattern setup $\bigtimes_{i\in I} \mathbb{P}_i$ is a pattern multistructure if and only if for all $i \in I$, pattern setup $\mathbb{P}_i$ is a pattern multistructure.

Proof. Let $A \in \mathcal{G}$. Description $(d_i)_{i\in I} \in cov(A)$ has a description $(c_i)_{i\in I} \in cov^*(A)$ such that $(d_i)_{i\in I} \sqsubseteq (c_i)_{i\in I}$ iff $(\forall i \in I)\ d_i \sqsubseteq c_i$. Recalling that $cov^*(A) = \bigtimes_{i\in I} cov_i^*(A)$ concludes the proof. ∎

3.8 Conclusion

In this chapter, we have provided an order-theoritic point-of-view on pattern languages. We have presented different frameworks stemming out from Formal Concept Analysis: (1) **Formal Concept Analysis** allows to handle directly Boolean datasets and provides the tool of conceptual scaling for handling more complex attributes. (2) More naturally, **Pattern Structures** model pattern languages as ordered sets but require that every set of objects must have a maximum common description. We have seen that this condition is rather too limiting to model some pattern search spaces as sequential ones [6]. (3) **Pattern Setups** model pattern languages just as posets without no additional properties making them too permissive. (4) The new framework of **Pattern Multistructures** lies between pattern setups and pattern structures and requires the weaker condition that the set of maximal common descriptions resumes properly the set of common descriptions of any subset of objects. Similarly to pattern structures that are tightly linked to (complete) lattices (Theorem 3.2), we have seen that pattern multistructures are tightly linked to (complete) meet-multisemilattices (Theorem 3.6).

While Pattern Multistructures provide many properties as the fact that support-closed descriptions is a correct condensed representation (Theorem 3.5), they still suffer from some issues as for instance the fact that the set of definable sets is not necessarily a join-multisemilattice (Proposition 3.9) or the fact that a projection of a pattern mutlistructure is not necessarily a pattern multistructure (Proposition 3.12). Another important result is that the usual antichain embedding used in the literature to build a pattern structure starting from a pattern setup (e.g. sequence of itemsets ones [39, 48]) is applicable if and only if the considered pattern setup is a pattern multistructure (Theorem 3.7). Still, one should be careful when using such a completion since the number of new definable sets in the completion could be exponential regarding the initial number of definable sets (see Example 3.31).

While the aim of this chapter is to understand the different properties a pattern language may provide, this chapter paves also the way to design algorithms to enumerate the set of definable sets of a pattern setup on a finite set of objects without completing it. The notion of enumeration will be investigated in the next chapters.

Part II

Pattern Enumeration Techniques

CHAPTER 4

SET SYSTEMS AND ENUMERATION

This chapter has for aim to present a small overview on the so-called **set systems**. A **set system** is formally any family of sets $\mathcal{F}$ on some ground set E, i.e. $\mathcal{F} \subseteq \wp(E)$. In fact, any partially ordered set $(P, \leq)$ can be embedded in $(\wp(P), \subseteq)$ via, for instance, the following order-embedding:

$$f : P \to \wp(P), p \mapsto {\downarrow} p = \{q \in P \mid q \leq p\}$$

This makes $(P, \leq)$ order-isomorphic to $(f[P], \subseteq)$. One can see that $(P, \leq)$ is a complete lattice iff $(f[P], \subseteq)$ is closed under arbitrary intersections. Other well-known relationship between finite set systems and finite distributive lattices is provided by the Birkhoff's Representation Theorem [25].

Understanding set systems is in general helpful to design efficient algorithms to **enumerate** particular elements of $\mathcal{F}$ as for instance closed elements of some closure operator $\phi : \mathcal{F} \to \mathcal{F}$. This chapter is inspired from [56], [29] and [130], and is organized as follows:

- **Section 4.1** presents basic notions on set systems and details the different properties that set systems can have. We study in detail at the end the particular set system built using the upper-ideals $\mathscr{U}(P)$ or lower-ideals $\mathscr{O}(P)$ on finite posets. Such a set system will be used in Chapter 5.
- **Section 4.2** presents particularly the notion of closure operator from set system perspective and how it does behave when we manipulate what is called convex geometries. An important structure that we will re-use in Chapter 5.
- **Section 4.3** presents the notion of enumeration problem and enumeration algorithm, a central notion in this dissertation for enumerating patterns. Particularly, we will be interested in enumerating closed elements in a set system. We study at the end two algorithms, namely `D&C` (subsection 4.3.3) and `ExR` (subsection 4.3.4).

4.1 Basic Definitions and Properties

Let us start by defining formally what a set system is.

Definition 4.1. Let E be a nonempty set and let $\mathcal{F} \subseteq \wp(E)$ be a nonempty family of sets. The pair $(E, \mathcal{F})$ is called a **set system** (or equivalently, a **hypergraph**) where E is called the **ground set** and elements of $\mathcal{F}$ are called **feasible sets** (or equivalently **hyperedges**). A set system $(E, \mathcal{F})$ is said to be **finite** if the ground set E is finite. An element $e \in E$ is said to be **isolated** if $e \notin \bigcup \mathcal{F}$, i.e. there is no feasible set enclosing it.

Note 4.1. Unless otherwise mentioned, we will consider here finite set systems $(E, \mathcal{F})$ with no isolated elements, that is $E = \bigcup \mathcal{F}$. Henceforth, $(E, \mathcal{F})$ could be abbreviated with its family of feasible sets $\mathcal{F}$. □

Example 4.1. Fig. 4.3 presents some set systems on the ground set $E = \{1,2,3,4\}$ with no isolated elements. □

Definition 4.2. The **dual set system** of a finite set system $\mathcal{F}$ is the set system $\mathcal{F}^\delta$ where:

$$\mathcal{F}^\delta := \{\bigcup \mathcal{F} \setminus S \mid S \in \mathcal{F}\}$$

Note 4.2. Please note that $\left(\mathcal{F}^\delta\right)^\delta = \mathcal{F}$. □

Example 4.2. In Fig. 4.3, set systems **(2)** and **(3)**, **(7)** and **(8)**; and **(10)** and **(11)** are dual set systems. □

4.1.1 On Set Systems Properties

We present now some properties on finite set systems. The terminology here follows [56] and [29].

Definition 4.3. A finite set system $\mathcal{F}$ (i.e. the ground set $E = \bigcup \mathcal{F}$) is said to have the
Accessibility property. if $\forall S \in \mathcal{F} \setminus \{\emptyset\}$, $\exists e \in S$ such that $S \setminus \{e\} \in \mathcal{F}$.
Smoothness property. if $\forall S, T \in \mathcal{F}$ s.t $S \subsetneq T$, $\exists e \in T \setminus S$ s.t. $S \cup \{e\} \in \mathcal{F}$.
Exchange property. if $\forall S, T \in \mathcal{F}$ s.t $|S| < |T|$, $\exists e \in T \setminus S$ s.t. $S \cup \{e\} \in \mathcal{F}$.
Transfer property. if $\forall S, T \in \mathcal{F}$ s.t $T \not\subseteq S$, $\exists e \in T \setminus S$ s.t. $S \cup \{e\} \in \mathcal{F}$.
Hereditary property. if $(\forall S \in \mathcal{F})\ T \subseteq S \Rightarrow T \in \mathcal{F}$.
Extendability property. if $\forall S \in \mathcal{F} \setminus \{E\}$, $\exists e \in E \setminus S$ such that $S \cup \{e\} \in \mathcal{F}$.
Closed-under-union property. if $(\forall S, T \in \mathcal{F})\ S \cup T \in \mathcal{F}$ and $\emptyset \in \mathcal{F}$.
Closed-under-intersection property. if $(\forall S, T \in \mathcal{F})\ S \cap T \in \mathcal{F}$ and $E \in \mathcal{F}$.

Before going deeper into other properties of set systems, we start by analyzing the implications between these different properties. All the implications that we prove in the following part of this section are synthesized in Fig. 4.1. Please recall that the set systems here are finite (i.e. there ground sets are finite). These implications do not necessarily hold for the case of infinite set

systems. Moreover, Exchange property definition does not have a meaning when infinite sets are considered. Let $(E, \mathcal{F})$ be a finite set system with $E = \bigcup \mathcal{F}$.

Proposition 4.1. We have:

$$\begin{aligned} \text{transfer property} &\Longrightarrow \text{exchange property} \\ \text{exchange property} &\Longrightarrow \text{smoothness property} \\ \text{hereditary property} &\Longrightarrow \text{smoothness property} \end{aligned}$$

Proof. The proof is straightforward for the two first implications since $|S| < |T| \Rightarrow T \not\subseteq S$ and $S \subsetneq T \Rightarrow |S| < |T|$. For the third implication, suppose that $S \subsetneq T$, we have $T \backslash S \neq \emptyset$. Moreover, $\forall e \in T \backslash S$ we have $S \cup \{e\} \subseteq T$. By Hereditary property we obtain $S \cup \{e\} \in \mathcal{F}$. ∎

Proposition 4.2. We have:

$$\text{transfer property } \textbf{and} \text{ the emptyset is feasible} \Longrightarrow \text{closed-under-union property}$$

Proof. Let be $(E, \mathcal{F})$ a finite set system where $\emptyset \in \mathcal{F}$ and that has the Transfer property and let $S, T \in \mathcal{F}$. It is clear that if $|T \backslash S| = 0$ (i.e. $T \subseteq S$) then $S \cup T = S \in \mathcal{F}$. Suppose now that $T \not\subseteq S$, if $|T \backslash S| = 1$ (i.e. $T \backslash S = \{e\}$) then since the set system has the antimatroid, we have $S \cup T = S \cup \{e\} \in \mathcal{F}$.

For a natural $n \geq 2$, if $|T \backslash S| = n$, there exist $e \in T \backslash S$ s.t. $S_1 = S \cup \{e\} \in \mathcal{F}$. It is clear that $|T \backslash S_1| = n - 1$. One can follow the same reasoning until showing that $T \cup S \in \mathcal{F}$. ∎

Proposition 4.3. We have:

$$\begin{aligned} \text{accessiblity property} &\Longrightarrow \emptyset \text{ is feasible} \\ \text{extendibilty property} &\Longrightarrow E \text{ is feasible} \end{aligned}$$

Proof. Let $S \in \mathcal{F}$, $|S| = n$ for some natural number $n \geq 1$. Using accessibility property, there exist $e \in S$ s.t. $S_1 = S \backslash \{e\} \in \mathcal{F}$ (i.e. $|S_1| = n - 1$). The same reasoning can be followed by induction until $n = 0$ (i.e. $\emptyset \in \mathcal{F}$). One can follow analogously the same steps to prove that if $\mathcal{F}$ is extendable then $E \in \mathcal{F}$ ∎

It is clear that the set system $(E, \mathcal{F})$ with $E = \{e_1, e_2\}$ and $\mathcal{F} = \{\{e_1\}, \{e_1, e_2\}\}$ has the smoothness property but not accessible since $\{e_1\} \neq \emptyset$ but yet there is no element in it that we can remove to create a new feasible set. The following proposition link both notions.

Proposition 4.4. Let $(E, \mathcal{F})$ be a finite set system with $E = \bigcup \mathcal{F}$ and having the smoothness property. We have:

- $(E, \mathcal{F})$ has accessibility if and only if $\emptyset \in \mathcal{F}$.
- $(E, \mathcal{F})$ has extendability if and only if $E \in \mathcal{F}$.

Proof. We have seen that $\emptyset \in \mathcal{F}$ is a necessary condition for accessibility. Let us show now that this condition is a sufficient one for the case of smooth set systems. Let $T \in \mathcal{F}$, since we have $\emptyset \in \mathcal{F}$, then, by using the smoothness property, there exists $e_1 \in T$ such that $T_1 = \emptyset \cup \{e_1\} \in \mathcal{F}$ (i.e. $|T_1| = 1$). If $T_1 = T$ then the proof is done. Otherwise, one can continue by creating feasible sets T_0, T_1, T_2, ..., $T_{|T|-1}, T_{|T|}$ with $T_{|T|} = T$, $T_0 = \emptyset$ and for all $1 \leq i \leq |T|$: $T_{i-1} = T_i \setminus \{e_i\}$. Hence, $\exists e_{|T|} \in T$ s.t. $T_{|T|-1} = T \setminus \{e_{|T|}\} \in \mathcal{F}$.

The second part of the proposition is trivial, i.e. use proposition 4.3 and the definition 4.3 for smoothness where $T = E$ and S is an arbitrary feasible set. ∎

Proposition 4.5. We have:

accessiblity property **and** closed-under-union property $\Longrightarrow$ transfer property

Proof. Let $S, T \in \mathcal{F}$ s.t. $T \not\subseteq S$. We need to show that $\exists e \in T \setminus S$ s.t. $S \cup \{e\} \in \mathcal{F}$. Using accessibility, one can create feasible sets by removing successively elements e from T until creating some feasible set $T' \subseteq T$ s.t. $|T' \setminus S| = 1$ (i.e. $T' \setminus S = \{e\}$). Since the set system is closed under union then $T' \cup S = S \cup \{e\} \in \mathcal{F}$. ∎

Proposition 4.6. We have:

extendability property **and** closed-under-intersection property $\Longrightarrow$ smoothness property

Proof. Let $S, T \in \mathcal{F}$ s.t. $S \subsetneq T$, we need to show that $\exists e \in T \setminus S$ such that $S \cup \{e\} \in \mathcal{F}$. Using the extendability of $\mathcal{F}$, one can create feasible sets by adding successively elements to S until creating some feasible set $S' \supseteq S$ s.t. $|S' \cap T| = |S| + 1$ (i.e. $S' \cap T = S \cup \{e\}$). Using the fact that $\mathcal{F}$ is closed-under-intersection and $S', T \in \mathcal{F}$ we conclude that $S \cup \{e\} \in \mathcal{F}$. ∎

Two properties P1 and P2 on set systems are said to be **dual properties** iff a set system has property P1 iff its dual set system has property P2 and *vice-versa* (See Definition 4.2).

Proposition 4.7. For a finite set system $(E, \mathcal{F})$ with $E = \bigcup \mathcal{F}$, the following pairs of properties are dual:

- $\emptyset \in \mathcal{F}$ **and** $E \in \mathcal{F}$.
- accessible **and** extendable.
- closed-under-intersection **and** closed-under-union.

Proof. Recall that $\mathcal{F}^\delta = \{E \setminus S \mid S \in \mathcal{F}\}$, we show below the three statements:

- $\emptyset \in \mathcal{F}$, then $E \setminus \emptyset \in \mathcal{F}^\delta$ and vice-versa.
- Suppose that $\mathcal{F}$ is accessible. Let $S \in \mathcal{F}^\delta \setminus \{E\}$. We have $E \setminus S \in \mathcal{F}$. Since $\mathcal{F}$ is accessible and $E \setminus S \neq \emptyset$ we have $\exists e \in E \setminus S$ s.t. $(E \setminus S) \setminus \{e\} = E \setminus (S \cup \{e\}) \in \mathcal{F}$. Therefore, $\exists e \in E \setminus S$ s.t. $S \cup \{e\} \in \mathcal{F}^\delta$, i.e. $\mathcal{F}^\delta$ is extendable.
- Suppose that $\mathcal{F}$ is closed under union. We have $\emptyset \in \mathcal{F}$ then $E \in \mathcal{F}^\delta$. Moreover, let $S, T \in \mathcal{F}^\delta$, we have $E \setminus S \in \mathcal{F}$ and $E \setminus T \in \mathcal{F}$, thus $(E \setminus S) \cup (E \setminus T) \in \mathcal{F}$. Hence, $(E \setminus (S \cap T)) \in \mathcal{F}$. Thus $S \cap T \in \mathcal{F}^\delta$. In other words, $\mathcal{F}^\delta$ is closed-under-intersection.

Fig. 4.1 synthesizes the different relationships between set systems properties for the case of finite set systems $(E, \mathcal{F})$ with $E = \bigcup \mathcal{F}$.

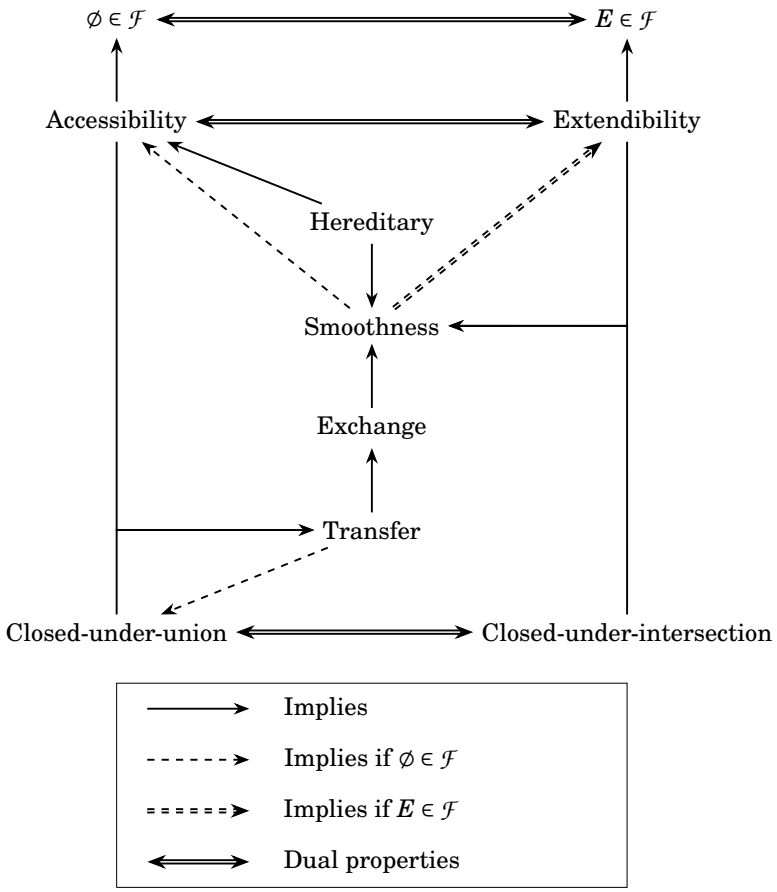


Figure 4.1: Implications between properties of finite set systems.

4.1.2 On Set Systems Structures

Definition 4.4. A finite set system $\mathcal{F}$ is said to be:

Accessible if it has the *accessibility property*.

Extendable if it has the *extendability property*.

Strongly Accessible if $\emptyset \in \mathcal{F}$ and it has *smoothness property*[1].

an Independence System if it has the *hereditary property*. In such case, elements of $\mathcal{F}$ are said to be **independent sets**. Otherwise, they are said to be **dependent**.

a Greedoid if $\emptyset \in \mathcal{F}$ and it has the *exchange property*.

a Matroid if it has the *exchange* and the *hereditary* property.

an Antimatroid if it is *closed under union* and is *accessible*.

a Convex Geometry if it is *closed under intersection* and is *extendable*.

Distributive if is is *closed both under intersection and union*.

Fig. 4.2 synthesizes the different relationships between finite set systems structures. One should notice that according to the different implication between properties that:

- A finite set system $\mathcal{F}$ is an antimatroid iff $\emptyset \in \mathcal{F}$ and has the *transfer property*.
- A finite set system $\mathcal{F}$ is an antimatroid iff its dual set system $\mathcal{F}^\delta$ is a convex geometry.
- A finite convex geometry is strongly accessible.
- A finite set system is distributive and accessible iff it is a convex geometry and an antimatroid.

Example 4.3. Fig. 4.3 give various example on set systems on $\{1,2,3,4\}$. We invite the reader to check the properties of each set system for the purpose of familiarization. □

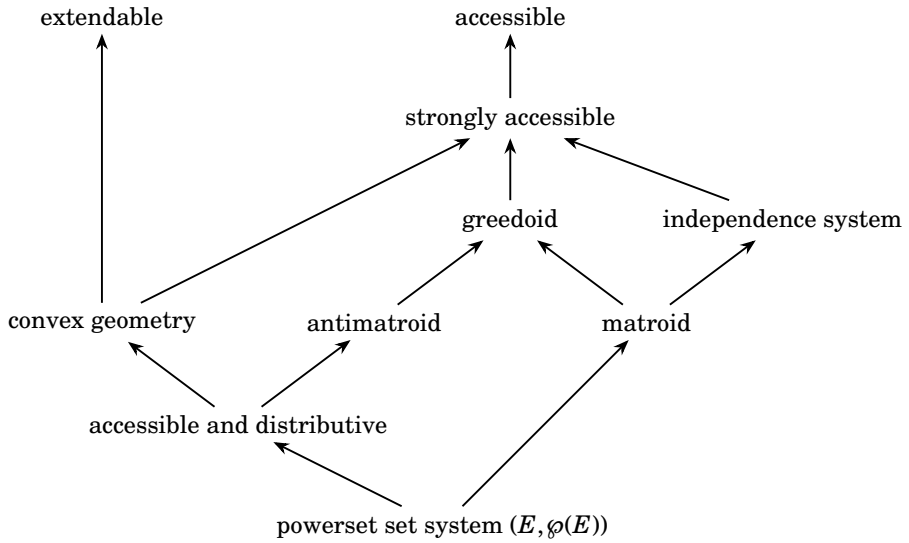


Figure 4.2: Structures on finite set systems $(E, \mathcal{F})$ with $E = \bigcup \mathcal{F}$ and their relationships

[1] The term *"strong accessibility"* was introduced in [28].

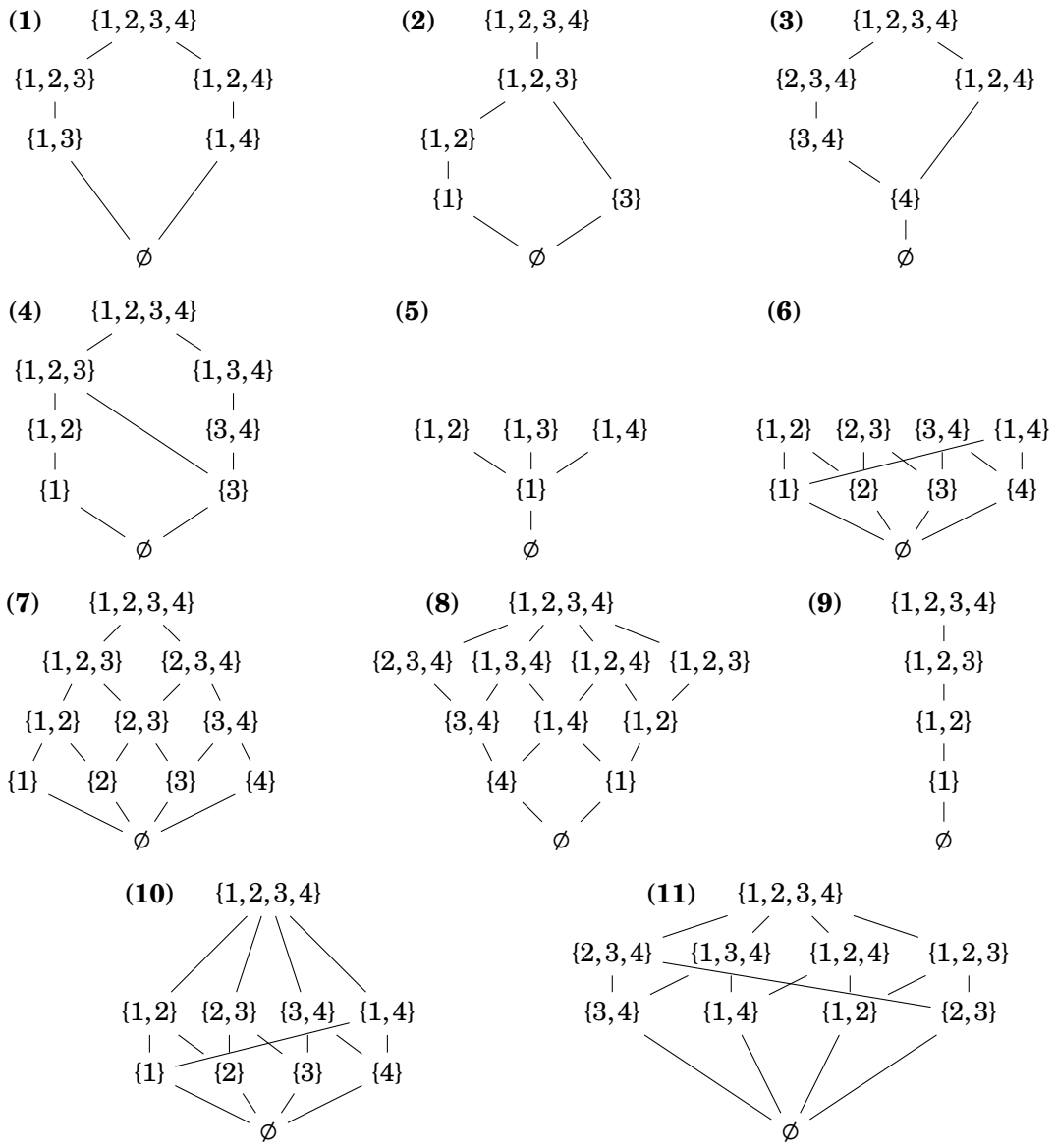


Figure 4.3: Some set systems on $\{1,2,3,4\}$: **(1)** A set system that is neither accessible nor extendable. **(2)** A set system that is accessible but not extendable. **(3)** A set system that is extendable but not accessible. **(4)** A set system that is accessible and extendable but not strongly accessible (consider $S_1 = \{1\}$ and $S_2 = \{1,2,3\}$)). **(5)** A greedoid but not an independence system since not all subsets of $\{1,4\}$ are into the set system. **(6)** A matroid. **(7)** A convex geometry. It does represent the set of all intervals of poset $(\{1,2,3,4\},\leq)$. **(8)** An antimatroid. **(9)** An accessible and distributive set system, it does represents the set of lower ideals of poset $(\{1,2,3,4\},\leq)$. **(10)** A set system that is closed under intersection but neither accessible nor extendable. **(11)** A set system that is closed under union but neither accessible nor extendable. Notice that set systems **(2)** and **(3)**, **(7)** and **(8)**; and **(10)** and **(11)** are dual set systems.

4.1.3 Lower and Upper Ideals over Finite Posets as Set Systems

In this section, our purpose is to study the particular set system of upper-ideals $\mathscr{U}(P)$ built over an arbitrary finite poset $(P,\leq)$. The aim of the section is to show that set system $\mathscr{U}(P)$ is distributive and accessible. Let us start by the following lemma.

Lemma 4.1. Let $(P,\leq)$ be a finite poset, we have:

$$(\forall S \in \mathscr{U}(P), \forall a \in P\backslash S)\ S \cup \{a\} \in \mathscr{U}(P) \quad \textbf{iff} \quad a \in max(P\backslash S)$$

Proof. Let $S \in \mathscr{U}(P)$ and let $a \in P\backslash S$. We show below both implications:

- ($\Leftarrow$) Let $a \in max(P\backslash S)$, that is: $(\uparrow a) \cap (P\backslash S) = \{a\}$. Hence, $\uparrow a \subseteq S \cup \{a\}$. We have $\uparrow (S \cup \{a\}) = \uparrow S \cup \uparrow \{a\} = S \cup \uparrow \{a\}$. Since, $\uparrow \{a\} \subseteq S \cup \{a\}$, then $S \cup \uparrow \{a\} \subseteq S \cup \{a\}$. By extensivity of $\uparrow$ we have $S \cup \{a\} \subseteq S \cup \uparrow \{a\}$. Thus, $\uparrow (S \cup \{a\}) = S \cup \{a\}$ or in other words $S \cup \{a\} \in \mathscr{U}(P)$.
- ($\Rightarrow$) Let $a \notin max(P\backslash S)$, that is $\exists b \in P\backslash S$ such that $a \leq b$ and $a \neq b$. Hence, $b \in \uparrow (S \cup \{a\})$ (since $b \in \uparrow a$) but in the same time $b \notin S \cup \{a\}$. In other words $\uparrow (S \cup \{a\}) \neq S \cup \{a\}$. Hence, $S \cup \{a\} \notin \mathscr{U}(P)$.

This concludes the proof. ∎

Proposition 4.8. For any finite poset $(P,\leq)$, the set system of upper ideals $\mathscr{U}(P)$ is *distributive* and *accessible*.

Proof. According to Note 2.22, we do have the fact that set system $\mathscr{U}(P)$ is *distributive*. Let us show that it has the transfer property (i.e. stronger than accessibility since $\emptyset \in \mathscr{U}(P)$).

Let be two upsets $A, C \in \mathscr{U}(P)$ s.t. $C \not\subseteq A$, we need to show that: $\exists a \in C\backslash A$ s.t. $A \cup \{a\} \in \mathscr{U}(P)$. Let us show before that $max(C\backslash A) \subseteq max(P\backslash A)$. Let $e \in max(C\backslash A)$, hence $e \in P\backslash A$. Let $f \in P$ s.t. $e \leq f$ and $e \neq f$. In one hand, since $e \in C$ we have $f \in C$ since $C \in \mathscr{U}(P)$. In the other hand, since $e \in max(C\backslash A)$ then $f \notin C\backslash A$. Therefore, $f \in A$ or in other words $f \notin P\backslash A$. We conclude that $e \in max(P\backslash A)$. Thus $max(C\backslash A) \subseteq max(P\backslash A)$.

According to Lemma 4.1, $\forall a \in max(P\backslash A)$, we have $A \cup \{a\} \in \mathscr{U}(P)$. Moreover, we have $max(C\backslash A) \neq \emptyset$ since P is finite and $C\backslash A \neq \emptyset$ (since $C \not\subseteq A$). Since $\emptyset \neq max(C\backslash A) \subseteq max(P\backslash A)$ then $\exists a \in C\backslash A$ s.t. $A \cup \{a\} \in \mathscr{U}(P)$. ∎

Note 4.3. Dual statements can be shown for set system $\mathscr{O}(P)$. □

4.2 Closure Operator on Finite Set Systems

According to Proposition 2.6 and Corollary 2.1, we know that a finite set system $(E, \mathcal{F})$ with $E = \bigcup \mathcal{F}$ is closed-under-intersection if and only if there is a closure operator $\phi_{\mathcal{F}} : \wp(E) \to \wp(E)$ for which we have $\phi_{\mathcal{F}}[\wp(E)] = \mathcal{F}$. Formally:

$$\phi_{\mathcal{F}} : \wp(E) \to \wp(E), A \mapsto \bigcap \{S \in \mathcal{F} \mid A \subseteq S\} \tag{4.1}$$

That is, $\phi_{\mathcal{F}}$ takes each subset in B to the smallest subset in $\mathcal{F}$ enclosing it. Hence, there is a characterization of set systems closed-under-intersection through closure operators.

From now on, $\mathcal{F}$ denotes a finite set system closed-under-intersection with no isolated elements (i.e. $E := \bigcup \mathcal{F}$) and $\phi_{\mathcal{F}}$ denotes its associated closure operator. The aim of this section is to study the properties of a closure operator associated to a closure system from a set system perspective.

Definition 4.5. Let $S \in \mathcal{F}$ be a closed set. A subset $B \subseteq S$ is said to be a **basis** or a **generator** of S iff: $\phi_{\mathcal{F}}(B) = S$. A basis $B \in \wp(E)$ is said to be **minimal** iff:

$$(\forall B' \subsetneq B)\ \phi_{\mathcal{F}}\left(B'\right) \subsetneq \phi_{\mathcal{F}}(B)$$

An interesting property identified by [151] and was crucial for designing an enumeration algorithm is formulated in the following proposition.

Proposition 4.9. If B is a *minimal basis* then all subsets of B are also *minimal basis*.

Proof. Let B be a minimal basis, i.e.:

$$(\forall B' \subsetneq B)\ \phi_{\mathcal{F}}\left(B'\right) \subsetneq \phi_{\mathcal{F}}(B)$$

If $B = \emptyset$ the proof ends here. Suppose now that $B \neq \emptyset$ and suppose that $\exists C \subsetneq B$ s.t. C is not a minimal basis. In other words:

$$(\exists D \subsetneq C)\ \phi_{\mathcal{F}}(D) = \phi_{\mathcal{F}}(B)$$

Let now be $B' = (B \backslash C) \cup D \subsetneq B$. Since $\phi_{\mathcal{F}}$ is order-preserving then $\phi_{\mathcal{F}}(B') \subseteq \phi_{\mathcal{F}}(B)$.

On the other hand, we have $D \subseteq B'$. Hence, $\phi_{\mathcal{F}}(D) = \phi_{\mathcal{F}}(C) \subseteq \phi_{\mathcal{F}}(B')$ since $\phi_{\mathcal{F}}$ is order-preserving. By extensivity, we conclude that $C \subseteq \phi_{\mathcal{F}}(B')$. Since $(B \backslash C) \subseteq \phi_{\mathcal{F}}(B')$ we conclude that $B \subseteq \phi_{\mathcal{F}}(B')$. Therefore, by idempotence and monotonicity we obtain $\phi_{\mathcal{F}}(B) \subseteq \mathcal{F}(B')$. Hence, $\phi_{\mathcal{F}}(B) = \phi_{\mathcal{F}}(B')$. This final statement is contradictory with the fact that B is a minimal basis. ∎

Note 4.4. According to Proposition 4.9, the set of minimal generators related to $\phi_{\mathcal{F}}$ form an *independent set system*. □

Another property identified by [153] shows also that one can build all minimal basis for some closed element using its lower neighbors fix-points thanks to minimal transversals [24]. Another important notion related to closure operator in set system is the notion of extreme points.

Definition 4.6. Let $S \in \mathcal{F}$ be a closed set. An element $e \in S$ is said to be an **extreme point** of S iff: $e \notin \phi_{\mathcal{F}}(S \backslash \{e\})$. The set of all extreme points of S is denoted $ex(S)$.

Proposition 4.10. Let $S \in \mathcal{F}$ be a closed set and let $e \in S$. We have:

$$S \backslash \{e\} \in \mathcal{F} \iff e \in ex(S)$$

Proof. We show both implications below:

($\Rightarrow$) Suppose that for $e \in S$ we have $S \backslash \{e\} \in \mathcal{F}$. Hence, $\phi_{\mathcal{F}}(S \backslash \{e\}) = S \backslash \{e\}$. Therefore, $e \notin \phi_{\mathcal{F}}(S \backslash \{e\})$, i.e. $e \in ex(S)$.

($\Leftarrow$) Let e be an extreme point, we need to show that $S \backslash \{e\} \in \mathcal{F}$. We have $S \backslash \{e\} \subseteq S$. Hence, $\phi_{\mathcal{F}}(S \backslash \{e\}) \subseteq \phi_{\mathcal{F}}(S) = S$. However, since $e \notin \phi_{\mathcal{F}}(S \backslash \{e\})$ then $\phi_{\mathcal{F}}(S \backslash \{e\}) \subsetneq S$. Since $\phi_{\mathcal{F}}$ is extensive we obtain $\phi_{\mathcal{F}}(S \backslash \{e\}) = S \backslash \{e\}$. In other words, $S \backslash \{e\} \in \mathcal{F}$.

This concludes the demonstration. ∎

Extreme points have a tight relationship with basis as formulated in the proposition below.

Proposition 4.11. Let $S \in \mathcal{F}$ and let $B \subseteq S$, if $\phi_{\mathcal{F}}(B) = S$ then $ex(S) \subseteq B$. In other words, all basis of S enclose $ex(S)$.

Proof. Let $B \subseteq S$ s.t. $\phi_{\mathcal{F}}(B) = S$ and let $e \in ex(S)$. Suppose that $e \notin B$, hence $B \subseteq S \backslash \{e\}$. Thus, using Proposition 4.10, we obtain $\phi_{\mathcal{F}}(B) \subseteq \phi_{\mathcal{F}}(S \backslash \{e\}) = S \backslash \{e\}$. Hence, $\phi_{\mathcal{F}}(B) \neq S$ which is a contradiction with the hypothesis that B is a basis of S. ∎

Another important properties on extreme points is the following.

Proposition 4.12. Let $S \in \mathcal{F}$ with non empty $ex(S)$ and let $e \in ex(S)$. We have:

$$ex(S) \backslash \{e\} \subseteq ex(S \backslash \{e\})$$

In other words, all extreme points remain extreme points after an extreme point removal.

Proof. If $ex(S) = \{e\}$, the proof ends here. Suppose now that $|ex(S)| > 1$ and let $e' \in ex(S) \backslash \{e\}$. We need to show that $e' \in ex(S \backslash \{e\})$. Since $S \backslash \{e'\}$ and $S \backslash \{e\}$ are in $\mathcal{F}$ and since set system $\mathcal{F}$ is closed under intersection we conclude that $S \backslash \{e, e'\} = S \backslash \{e\} \cap S \backslash \{e'\} \in \mathcal{F}$. Hence, according to Proposition 4.10 ($\Leftarrow$), we obtain $e' \in ex(S \backslash \{e\})$. ∎

When dealing with a finite convex geometry $\mathcal{F}$, the closure operator $\phi_{\mathcal{F}}$ is called a **convex hull** [158]. Interestingly, the **convex hull** has additional properties as detailed in [60]. Among them, we draw the reader's attention to the equivalent properties that characterize the closure operator of a convex geometry.

Theorem 4.1 (A subset of Theorem 2.1 in [60]). Let $\mathcal{F}$ be a finite set system that is closed under intersection with $\emptyset \in \mathcal{F}$. The following properties are equivalent:

- $\mathcal{F}$ is extendable, i.e. $\mathcal{F}$ is a convex geometry.
- For any $S \in \mathcal{F}$, S has a unique basis.
- For any $S \in \mathcal{F}$, $\phi_{\mathcal{F}}(ex(S)) = S$, i.e $ex(S)$ is the unique minimal basis of S.
- For any $S \in \mathcal{F}$ and $e \notin S$ then $e \in ex(\phi_{\mathcal{F}}(S \cup \{e\}))$.
- The closure operator $\phi_{\mathcal{F}}$ has the **anti-exchange property** given below (with $E = \bigcup \mathcal{F}$):

$$(\forall S \in \mathcal{F}, \forall e_1, e_2 \in E)\ e_1 \in \phi_{\mathcal{F}}(S \cup \{e_2\}) \Rightarrow e_2 \notin \phi_{\mathcal{F}}(S \cup \{e_1\})$$

Example 4.4. Consider the set system **(10)** depicted in Fig. 4.3 that we will denote here $\mathcal{F}$. It is easy to see that this set system is closed under intersection but not extendable, i.e. it is not a convex geometry. Clearly, according to Proposition 4.10, $\{1,2,3,4\}$ have no extreme points. Moreover, it has 4 distinct basis: $\{1,2,3\}$, $\{1,2,4\}$,$\{1,3,4\}$ and $\{2,3,4\}$ (i.e. use the associated closure operator $\phi_{\mathcal{F}}$ presented in eq. 4.1 to check if these sets are basis).

Consider now the convex geometry depicted in Fig. 4.3 **(7)**. One can see that extreme points of each set is the minimum and the maximum w.r.t. $\leq$. For instance, $ex(\{1,2,3,4\}) = \{1,4\}$ which is the unique basis of $\{1,2,3,4\}$. □

Before leaving this section, let us reconsider the set system of upper-ideals and lower-ideals of a finite poset $(P, \leq)$ studied in Section 4.1.3. Since these two set systems $\mathscr{U}(P)$ and $\mathscr{O}(P)$ are distributive and accessible then they are convex geometries (cf. Fig. 4.2). Therefore, the associated closure operator $\uparrow$ and $\downarrow$ are *convex hulls*. Moreover, one can show the following fact:

$$(\forall S \in \mathscr{U}(P))\ ex_{\mathscr{U}(P)}(S) = min(S) \quad \textbf{and} \quad (\forall S \in \mathscr{O}(P))\ ex_{\mathscr{O}(P)}(S) = max(S)$$

Where $ex_{\mathscr{U}(P)}$ (resp. $ex_{\mathscr{O}(P)}$) denotes the extreme points mapping in set system $\mathscr{U}(P)$ (resp. $\mathscr{O}(P)$).

4.3 Enumeration Algorithms on Set Systems

In this section, we will be interested particularly in the enumeration of fixpoints of a closure operator on an arbitrary finite set system (see Problem 4.1). We take the opportunity in this section, to explain "enumeration" problems and the different notions around it. Next, we will present two important algorithms solving Problem 4.1, two algorithms that will be useful next in this dissertation. For more details about enumeration problems, please refer to the thesis [130] which presents the different notions about enumerations problems and algorithms in a didactic way.

4.3.1 Enumeration Problem and Algorithms

Let us start by defining formally in a general way what is an (enumeration) **problem** .

Definition 4.7. A **problem** is a triple (P,S,R) where P is an arbitrary set called **problem sets**, S is an arbitrary set called the **solution set** and $R \subseteq P \times S$ be a binary relation over P and S. We say that an element $s \in S$ **solves** a problem instance $p \in P$ iff $p\ R\ s$. For each problem $p \in P$, we denote by $R(p)$ the set of solutions of p. It is given by: $R(p) := \{s \in S \mid p\ R\ s\}$. We state the additional condition that for every $p \in P$, the set of its solutions $R(p)$ is **finite**.

Example 4.5. Consider for instance the problem of finding maximal cliques of an undirected graphs. The set P is the set of all possible finite undirected graphs (V,E) and the set of solutions S are finite sets. An element $s \in S$ is a solution of a problem instance (V,E) if $s \subseteq V$ and s is the set of vertex of a maximal clique on (V,E). □

Following Definition 4.7, an **enumeration problem** is the task of **listing** for each problem instance $p \in P$ its set of solutions $R(p)$ (which is finite). To solve an enumeration problem, one should build a **listing algorithm**. In a nutshell, a listing algorithm could also be seen as a mapping $\mathscr{A}$ that associates to each instance $p \in P$ a sequence $\mathscr{A}(p)$ of outputs on S. A sequence of outputs $\mathscr{A}(p)$ can be seen as a pair (n,s) where n is the size of sequence and $s : \{1,...,n\} \to S$ with $s(i)$ designates the i^{th} element. Definition 4.8 states three properties that a listing algorithm should have to solve an enumeration problem.

Definition 4.8. Given a problem (P,S,R). The listing algorithm $\mathscr{A}$ is said to be:

- **Sound:** if $\forall p \in P$, algorithm $\mathscr{A}$ outputs only solutions in $R(p)$. In other words, for the sequence of outputs $\mathscr{A}(p) := (n,s)$, we have $s[\{1,...,n\}] \subseteq R(p)$.
- **Complete or Exhaustive:** if $\forall p \in P$, all solutions in $R(p)$ are output at least once. In other words, for the sequence of outputs $\mathscr{A}(p) := (n,s)$, we have $s[\{1,...,n\}] \supseteq R(p)$.
- **Non-redundant:** if $\forall p \in P$, all outputs of $\mathscr{A}$ are output exactly once. In other words, for the sequence of outputs $\mathscr{A}(p) := (n,s)$, we have s injective, i.e. $|s[\{1,...,n\}]| = n$.

If the listing algorithm $\mathscr{A}$ verifies the three aforementioned properties, we say that it is a **correct enumeration algorithm** for the enumeration problem (P,S,R).

4.3.1.1 Enumeration algorithm time complexity

Given an enumeration problem (P,S,R), there are many *correct enumeration algorithms* $\mathscr{A}$ that solve the problem correctly. Yet, one should evaluate the complexity of the enumeration algorithm in order to compare them. In order to explain the notion of **time complexity**, suppose now that we have some function $size$ that associates to each $p \in P$ its size $size(p) \in \mathbb{N}^*$. It does also associate to each solution $s \in S$, a size $size(s) \in \mathbb{N}^*$. The size of the set of solutions $size(R(p))$ is then given by the sum of the sizes of each solution, i.e. $size(R(p)) = \sum_{s \in R(p)} size(s)$.

Example 4.6. Consider again the problem presented in Example 4.5. For an input instance $p := (V,E)$, the size could be given by $size(p) = |V| + |E| = O\left(|V|^2\right)$. The size of a solution $s \subseteq V$ is given by $size(s) = |s| = O(|V|)$. One should note however, that the number of solutions $|R(p)|$ can goes up to $3^{|V|/3}$ [131]. □

As shown in Example 4.6, the number of outputs of a correct enumeration algorithm could be exponentially larger than the size of its input. The complexity of an enumeration algorithm is evaluated as a function of the input and the output sizes (**output-sensitive complexity**) since the algorithm needs to output all the solutions anyway [102, 130].

We distinguish two *time complexities*:

- **The Total Complexity** gives a bound (using big O notation) on the total enumeration time, i.e. the instant when the algorithm announces it ends, as a function of the input and the output sizes.
- **The Delay Complexity** gives a bound on the delay between two successive outputs, the beginning of the algorithm and the first output or the last output of the algorithm and the end of the algorithm, as a function of the input size.

If the total complexity of algorithm $\mathscr{A}$ is polynomial to $size(p) + size(\mathcal{R}(p))$, we say that $\mathscr{A}$ is `POLY-OUTPUT`. If the delay complexity of algorithm $\mathscr{A}$ is polynomial to $size(p)$, we say that $\mathscr{A}$ is `POLY-DELAY`. Clearly, all `POLY-DELAY` algorithms are `POLY-OUTPUT`. There are other classes of complexity of enumeration algorithms that are left out of the scope of this thesis.

4.3.1.2 Enumeration algorithms space complexity

Beside the **time complexity**, the **space complexity** required by an enumeration algorithm $\mathscr{A}$ has also to be evaluated. the **space complexity** gives a bound on the total space that can be used by algorithm $\mathscr{A}$ at the same time as a function of $size(p) + size(\mathcal{R}(p))$.

An algorithm $\mathscr{A}$ is said to be `PSPACE` if its space complexity is polynomial to the input size $size(p)$. For instance, a basic algorithm that stores all solutions in order to ensure for example non-redundancy is rarely `PSPACE` since the output could be exponentially larger than the input (see Example 4.6).

4.3.2 Listing Fixpoints of a Closure Operator Problem

Now that we have understood what are the notions around enumeration problems and algorithms, we consider now the particular problem of enumerating fixpoints of a closure operator on a set system. We formulate the problem below.

Problem 4.1. Let $(E, \mathcal{F})$ be a finite set system with no isolated elements, i.e. $E = \bigcup \mathcal{F}$. Let σ be a closure operator on $(\mathcal{F}, \subseteq)$. List all fixpoints $\sigma[\mathcal{F}]$.

Note 4.5. Before diving deeper, please note that this problem is more generic than listing all sets of an arbitrary finite set system $\mathcal{F}$. Indeed, it suffices to consider the closure operator σ as the identity operator, i.e. $\sigma : \mathcal{F} \to \mathcal{F}, f \mapsto f$.

Please notice also that if $\mathcal{F} = \wp(E)$ then $\sigma[\mathcal{F}]$ is closed under intersection. However, this is not necessarily the case for an arbitrary set system $(E, \mathcal{F})$. For instance, if $(E, \mathcal{F})$ is not closed under intersection, the identity operator $id : \mathcal{F} \to \mathcal{F}$ is a closure operator which fixpoints $id[\mathcal{F}] = \mathcal{F}$. Hence, $id[\mathcal{F}]$ is not a Moore family. □

For a given enumeration algorithm solving Problem 4.1, we will often distinguish (informally) between two general types of traversal.

- **Bottom-up Traversal:** Such a traversal starts from the minimal elements in $\sigma[\mathcal{F}]$ (w.r.t $\subseteq$) then enumerates higher elements (w.r.t $\subseteq$). An algorithm doing such a traversal is presented in Section 4.3.3.
- **Top-down Traversal:** Such a traversal starts from the maximal elements in $\sigma[\mathcal{F}]$ (w.r.t $\subseteq$) then enumerates smaller elements (w.r.t $\subseteq$). An algorithm doing such a traversal is presented in Section 4.3.4.

In fact, suppose that we have some constraints on $S \in \sigma[\mathcal{F}]$ s.t. if it does hold for S it does hold for all $T \in \sigma[\mathcal{F}]$ s.t. $T \subseteq S$ and we want to output solely elements $\sigma[\mathcal{F}]$ for which the constraint hold. For instance, the constraint $|S| \leq n$ with $n \in \mathbb{N}$ verifies such a property. Algorithms with a Bottom-up traversal can be adapted easily to outputs solely elements $\sigma[\mathcal{F}]$ for which the constraint hold. Indeed, whenever a subset $T \in \sigma[\mathcal{F}]$ does not verify the constraint, one should not visit its super-sets. Dual remark can be done for *top-down traversal*.

4.3.3 `D&C` Algorithm - An Algorithm for Strongly Accessible Set Systems

Algorithm 1, dubbed *Divide & Conquer Closed Set Listing* (`D&C` for short), tackles Problem 4.1. It was shown that Algorithm 1 is *correct* if and only if the considered set system $(E, \mathcal{F})$ is finite and strongly accessible [29], i.e. it solves the following subproblem of Problem 4.1.

Problem 4.2. Let $(E, \mathcal{F})$ be a finite strongly accessible set system with no isolated elements, i.e. $E = \bigcup \mathcal{F}$. Let σ be a closure operator on $(\mathcal{F}, \subseteq)$. List all fixpoints $\sigma[\mathcal{F}]$.

One should note that this algorithm draws its roots from various works in the litterature that solves Problem 4.1 when the considered set system is the powerset set system ($i.e. \mathcal{F} := \wp(E)$).

Algorithm 1: D&C (Divide & Conquer Closed Set Listing) Algorithm

Input: Finite strongly accessible set system $(E, \mathcal{F})$ and a closure operator σ on $(\mathcal{F}, \subseteq)$.
Output: Elements of $\sigma[\mathcal{F}]$

```
1 procedure D&C(C, B)
2     for e ∈ E\(B ∪ C) s.t. C ∪ {e} ∈ F do
3         C_new ← σ(C ∪ {e})                  // Compute the new closed element C_new
4         if C_new ∩ B = ∅ then
5             D&C(C_new, B)
6         B ← B ∪ {e}                         // Update the set of banned elements B
7     Print(C)                                // Output the closed element C ∈ σ[F]
8 D&C(σ(∅), ∅)                                // Enumeration starts here on σ(∅)
```

One can cite [33, 76, 109, 110] among others. It was however Gély [83] who showed that many of those algorithms can be seen as an instance of a more general *divide-&-conquer* which was revisited then by [29] to enlarge it scope to finite strongly accessible set system rather that only the trivial set systems (i.e. powersets).

Algorithm 1 performs a bottom-up and depth-first traversal of $\sigma[\mathcal{F}]$. It starts from the smallest element $\sigma(\emptyset)$ in $\sigma[\mathcal{F}]$ (Line 8) then enumerates in depth-first fashion elements in $\sigma[\mathcal{F}]$ by performing closure computations (Line 3) then checking, thanks to *canonicity test* (Line 4), if the closed set is already generated making D&C non-redundant. We gave here a simpler version of Algorithm 1, a memory efficient algorithm equivalent to Algorithm 1 in its behavior has been also presented in [29].

Theorem 4.2. Algorithm 1 solves the problem of the exhaustive and non-redundant enumeration of the fixpoints of the closure operator σ on a finite strongly accessible set system $\mathcal{F}$ with the following complexities:

$$\begin{aligned} &\textit{delay } O\big(|E| \cdot \big(T_\sigma + T_\mathcal{F} + |E|\big)\big), \text{ and} \\ &\textit{total } O\big(|E| \cdot \big(T_\sigma + T_\mathcal{F} + |E|\big) \cdot |\sigma(\mathcal{F})|\big), \text{ and} \\ &\textit{space } O\big(|E|^2 + S_\sigma + S_\mathcal{F}\big) \end{aligned}$$

Where T_σ ($T_\mathcal{F}$) and S_σ ($S_\mathcal{F}$) are respectively the time and space complexity of computing the closure (checking if an element is in $\mathcal{F}$).

Proof. To see the delay complexity, one should notice that the algorithm cannot backtrack more that $|E|$ times without printing an element. Each call of D&C costs one closure computation T_σ, one membership checking $T_\mathcal{F}$ and manipulation (intersection, union) of sets of size $O(|E|)$. The formula of delay complexity follows. The total complexity is directly deducted from the delay complexity. For the space complexity follow also from the fact that there is no more than $|E|+1$ nested call of D&C algorithm. Notice that in each call D&C manipulates a constant set of variable of size $O(|E|)$. ∎

Note 4.6. One should note that the *total complexity* is not always equal to the *delay complexity* multiplied by the number of solutions, i.e. the total complexity could be better. For instance, Sergei O. Kuznetsov attracted our attention to the (fun) fact that if Algorithm D&C outputs the closed element before the loop, its delay complexity becomes $O\big(|E|^2 \cdot \big(T_\sigma + T_\mathcal{F} + |E|\big)\big)$ while the total complexity remains obviously the same. □

4.3.4 ExR Algorithm - An Algorithm for Convex Geometries

We present here a second algorithm in order to show that when the closure operator has additional properties, someone need to leverage them in order to enumerate its fixpoints more efficiently. We consider here for instance the following subproblem of Problem 4.1.

Problem 4.3. Let E be a finite set and let σ be a closure operator on $(\wp(E), \subseteq)$ having the **anti-exchange property**:

$$(\forall S \in \sigma[\wp(E)], \forall e_1, e_2 \in E)\ e_1 \in \sigma(S \cup \{e_2\}) \Rightarrow e_2 \notin \sigma(S \cup \{e_1\})$$

List all fixpoints $\sigma[\wp(E)]$.

Note 4.7. Please note that according to Theorem 4.1, Problem 4.3 is equivalent to enumerate all elements of a convex geometry $\mathcal{F}$. Indeed, set system $(E, \sigma[\wp(E)])$ is a convex geometry and any convex geometry can be induced by a closure operator having the anti-exchange property. □

Algorithm 2, namely Extreme points Removal Closed Set Listing or ExR for short, is a correct enumeration algorithm for Problem 4.3. It performs a top-down and depth-first traversal of $\sigma[\mathcal{F}]$. It starts from the top element E which is always closed by extensivity of σ (Line 7) then enumerates in depth-first fashion the convex sets in $\sigma[\wp E]$. Rather than performing closures, Algorithm 2 relies on Proposition 4.10, i.e. to create new closed set, it does successive extreme points removal (Line 4). To ensure non-redundancy, Algorithm 2 maintains a set of banned elements (Line 6), i.e. whenever an extreme point is removed from a closed set C, it is forbidden to remove it in the next iterations.

Algorithm 2: ExR (Extreme points Removal Closed Set Listing) Algorithm

Input: A finite set E and
a closure operator σ on $(\wp(E), \subseteq)$ having the **anti-exchange property**
Output: Elements of the convex geometry $\sigma[\wp(E)]$

```
1 procedure ExR(C, B)
2     Print(C)                                   // Output the convex set φ[℘(E)]
3     for e ∈ ex(C)\B do
4         C_new ← C\{e}          // generate a new convex set C_new since e ∈ ex(C)
5         ExR(C_new, B)
6         B ← B ∪ {e}      // Element e can not be removed in the next iterations
7 ExR(E, ∅)                                      // Enumeration starts here on E
```

Theorem 4.3 provide the computation and space complexities of Algorithm 2.

Theorem 4.3. Algorithm 2 solves Problem 4.3 with the following complexities:

$$\begin{aligned} &\textit{delay } O(|E|\cdot(T_{ex}+|E|)), \text{ and} \\ &\textit{total } O(|E|\cdot(T_{ex}+|E|)\cdot|\mathcal{F}|), \text{ and} \\ &\textit{space } O\left(|E|^2+S_{ex}\right) \end{aligned}$$

Where T_{ex} and S_{ex} are respectively the time and space complexity of computing the extreme points of a given convex set.

Proof. To see the delay complexity, one should notice that the algorithm cannot backtrack more that $|E|$ times without printing an element. Each call of ExR costs one extreme points computation T_{ex} and manipulation (union, difference) of sets of size $O(|E|)$. The formula of delay complexity follows. The total complexity is directly deducted from the delay complexity. For the space complexity follow also from the fact that there is no more than $|E|+1$ nested call of ExR algorithm. Notice that in each call ExR manipulates a constant set of variable of size $O(|E|)$, adding the space cost of extreme points computation S_{ex} at each call, the formula of space complexity follows..

CHAPTER 5

ENUMERATION IN PATTERN SETUPS

We have seen in Chapter 3 that pattern languages can be seen as ordered sets. Particularly, we considered the general case of pattern setups to model pattern languages over a dataset. The aim of this chapter is to investigate the following problem:

Problem 5.1. Let $\mathbb{P} = (\mathcal{G}, (\mathcal{D}, \sqsubseteq), \delta)$ be a finite pattern setup (i.e. set of objects $\mathcal{G}$ is finite). List **all** definable sets in $\mathbb{P}_{ext}$ **non-redundantly**.

It is clear that $\mathbb{P}_{ext}$ is a set system on $\mathcal{G}$. Moreover, when the considered pattern setup $\mathbb{P}$ is a pattern structure, the set system $\mathbb{P}_{ext}$ is closed-under-intersection which associated closure operator is $ext \circ int$. Therefore, we will often use here results from Chapter 4 to propose correct algorithms solving Problem 5.1. The following of this chapter is organized as follow:

- **Section 5.1** investigates a sub-problem of Problem 5.1 when the considered pattern setup is a pattern structure over itemsets (or equivalently a formal context).
- **Section 5.2** extends the state-of-the-art by proposing a new algorithm for enumerating definable sets of a formal context by leveraging the implications existing inherently between its attributes. This contribution had appeared in [16].
- **Section 5.3** considers the larger problem of enumerating definable sets of an arbitrary pattern structure.
- **Section 5.4** considers Problem 5.1 for the particular pattern structure of interval patterns in numerical datasets.
- **Section 5.5** considers the convex set pattern language and three algorithms solving Problem 5.1 for finite pattern structures induced by this language. The results presented in this section were introduced in [20].
- **Section 5.6** concludes this chapter by discussing Problem 5.1 for the general case of pattern setups. Recall that sequential patterns [6] does not induce a pattern structure [48].

5.1 Enumeration in Formal Contexts

In this section, we will consider Problem 5.1 where the inputs are restricted to formal contexts $\mathbb{P} = (\mathcal{G}, \mathcal{M}, I)$ with finite set of objects $\mathcal{G}$ and finite the set of attributes $\mathcal{M}$. All the notations we use here follow Section 3.2 on formal contexts. The new sub-problem is formulated below:

Problem 5.2. Let $\mathbb{P} = (\mathcal{G}, \mathcal{M}, I)$ be a formal context with a finite set of objects $\mathcal{G}$ and a finite the set of attributes $\mathcal{M}$. List **all** definable sets in $\mathbb{P}_{ext}$ **non-redundantly**.

Solving Problem 5.2 is tightly linked to enumerating fixpoints of a given closure operator that was thoroughly investigated in the literature. We have seen in Section 4.3.2 that many algorithms solving this later problem (e.g. [33, 76, 109, 110]) follows the same enumeration scheme, i.e. *divide-&-conquer* scheme, as stated by Gély [83] and later by [29] for a further generalization. The general Algorithm is dubbed `D&C` and is presented in Algorithm 1. We will present here two particular algorithms solving Problem 5.2 in formal contexts which is CLOSE-BY-ONE (CBO) [109, 110]. Both algorithms rely on the fact that the pair of operators (ext, int) form a Galois connection between posets $(\wp(\mathcal{G}), \sqsubseteq)$ and $(\wp(\mathcal{M}), \sqsubseteq)$ (Proposition 3.1). Hence, mappings $ext \circ int$ and $int \circ ext$ are closure operators on $(\wp(\mathcal{G}), \sqsubseteq)$ and $(\wp(\mathcal{M}), \sqsubseteq)$ respectively. Hence, one can rely on Algorithm 1 to enumerate all extents in $\mathbb{P}_{ext}$ using two different approaches.

- Algorithm CLOSE-BY-ONE BOTTOM-UP enumerate extents in a Depth-first search and bottom up fashion, i.e. from small extents to bigger ones. It uses the closure operator $ext \circ int$ on the powerset set system $\wp(\mathcal{G})$.
- Algorithm CLOSE-BY-ONE TOP-DOWN enumerate extents in a Depth-first search and top down fashion, i.e. from bigger extents to smaller ones. It uses the closure operator $int \circ ext$ on the powerset set system $\wp(\mathcal{M})$ and the fact that there is a one-to-one correspondance with the set of fixpoints of $int \circ ext$ given by $\mathbb{P}_{int}$ and the extents $\mathbb{P}_{ext}$, i.e. $\mathbb{P}_{ext} = ext[\mathbb{P}_{int}]$.

5.1.1 Algorithm CLOSE-BY-ONE BOTTOM-UP

Algorithm 3, dubbed CLOSE-BY-ONE BOTTOM-UP or CBO-BU for short, does enumerate elements of $\mathbb{P}_{ext}$ using Algorithm 1 where the considered set system is $(\mathcal{G}, \wp(\mathcal{G}))$ and $\sigma := ext \circ int$. Algorithm 3 is useful when someone wants to enumerate definable sets in increasing support. This allows for instance to look for *rare subgroups*/*patterns* efficiently [2, 152].

Theorem 5.1. Algorithm 3 solves Problem 5.2 with:

$$\begin{aligned} &\textit{delay } O\big(|\mathcal{G}|^2 \cdot |\mathcal{M}|\big), \text{ and} \\ &\textit{total } O\big(|\mathcal{G}|^2 \cdot |\mathcal{M}| \cdot \mathbb{P}_{ext}\big), \text{ and} \\ &\textit{space } O\big(|\mathcal{G}| \cdot \big(|\mathcal{G}| \cdot |\mathcal{M}|\big)\big) \end{aligned}$$

That is Algorithm 3 is `POLY-DELAY` and `PSPACE`.

Algorithm 3: Algorithm CLOSE-BY-ONE BOTTOM-UP (CBO-BU)

Input: $\mathbb{P} = (\mathcal{G}, \mathcal{M}, I)$ a formal context
Output: Elements of $\mathbb{P}_{ext}$

```
1 procedure CbO-BU(A, C, B)
2     for g ∈ 𝒢\(B ∪ A) do
3         C_new ← C ∩ int(g)                    // Compute the new intent C_new
4         A_new ← ext(C_new)                    // Compute the new extent A_new
5         if A_new ∩ B = ∅ then
6             CbO-BU(A_new, C_new, B)
7         B ← B ∪ {g}                           // Update the set of banned objects B
8     Print(A)                                  // Output extent A
9 CbO-BU(ext(ℳ), ℳ, ∅)
```

Proof. One can use directly Theorem 4.2 where $S_\sigma = T_\sigma = O\big(\mathcal{G} \cdot \mathcal{M}\big)$ and $T_\mathcal{F} = S_\mathcal{F} = O(1)$. ∎

5.1.2 Algorithm CLOSE-BY-ONE TOP-DOWN

Algorithm 4, dubbed CLOSE-BY-ONE TOP-DOWN or CBO-TD for short, enumerates elements of $\mathbb{P}_{ext}$ using Algorithm 1 where the considered set system is $(\mathcal{M}, \wp(\mathcal{M}))$ and $\sigma := int \circ ext$ but outputs at line 7 the extent of the closed elements. Conversely to CBO-BU, Algorithm CBO-TD enumerates the definable sets in decreasing support. Hence, this algorithm is fashioned for enumerating frequent patterns first since it is easy to integrate pruning strategies of order-reversing (anti-monotonic) quality measure.

Theorem 5.2. Algorithm 3 solves Problem 5.2 with:

$$\begin{aligned} &\textit{delay } O\big(|\mathcal{G}| \cdot |\mathcal{M}|^2\big), \text{ and} \\ &\textit{total } O\big(|\mathcal{G}| \cdot |\mathcal{M}|^2 \cdot \mathbb{P}_{ext}\big), \text{ and} \\ &\textit{space } O\big(|\mathcal{M}| \cdot \big(|\mathcal{G}| \cdot |\mathcal{M}|\big)\big) \end{aligned}$$

That is Algorithm 4 is POLY-DELAY and PSPACE.

Proof. One can use directly Theorem 4.2 where $S_\sigma = T_\sigma = O\big(\mathcal{G} \cdot \mathcal{M}\big)$ and $T_\mathcal{F} = S_\mathcal{F} = O(1)$. ∎

Note 5.1. We called here both algorithms CBO (CLOSE-BY-ONE) [109, 110]. In fact, the only difference between the algorithms presented here and the base CBO is the canonicity test (Line 5 in both algorithms) used to ensure the non-redundancy of the enumeration. The base canonicity test used in [109, 110] depends on the order introduced in [76]. That is rather than keeping a set of banned objects (resp. attributes), the set of objects (resp. attributes) is totally ordered and

Algorithm 4: Algorithm CLOSE-BY-ONE TOP-DOWN (CBO-TD)

Input: $\mathbb{P} = (\mathcal{G}, \mathcal{M}, I)$ a formal context
Output: Elements of $\mathbb{P}_{ext}$

```
1 procedure CbO-TD(A, C, B)
2     for m ∈ M\(B ∪ C) do
3         A_new ← A ∩ ext(m)                    // Compute the new extent A_new
4         C_new ← int(A_new)                    // Compute the new intent C_new
5         if C_new ∩ B = ∅ then
6             CbO-TD(A_new, C_new, B)
7         B ← B ∪ {m}                  // Update the set of banned attributes B
8     Print(A)                                              // Output extent A
9 CbO-TD(G, int(G), ∅)
```

whenever the i^{th} object (resp. attribute) is added, no object (resp. attribute) which index is strictly below i is allowed to be added to the newly generated set of objects (resp. attributes) after closure. We invite the reader to see [76] or Section 2.1 in [80] for more details. For a more memory efficient algorithm, please see Algorithm 2 in [29]. □

5.1.3 Conclusion

We have presented here two basic and simple, yet fast and efficient, algorithms to enumerate definable sets of a formal context. These algorithms can further be enhanced. One can think for instance about FAST CBO (FCBO) [107, 136] to reduce the number of canonicity test fail or the series of INCLOSE Algorithms [8]. Moreover, it is important to note that while these algorithms aim to enumerate exhaustively all extents of a formal context, they can be easily adapted to output only a subset of extents w.r.t. additional constraints. For instance, it is easy to integrate the constraint *"extent size is higher than* $minsup \in \mathbb{N}$*"* in Algorithm 4 or *"extent size is lower than* $maxsup \in \mathbb{N}$*"* in Algorithm 3. Other type of constraints have widely been investigated in the literature (see [32, 132]). Last but not least, while these algorithms are defined for datasets that come as formal contexts, one can consider a larger set of datasets with more complex attributes (e.g. numerical, categorical) thanks to conceptual scaling (see Section 3.2.3). In the perspective of enhancing these algorithms on scaled contexts, Section 5.2 presents a new algorithm, dubbed CBOI for CLOSE-BY-ONE USING IMPLICATIONS [16].

5.2 Enumeration in Formal Contexts using Implications

Formal Concept Analysis (FCA) provides a mathematical tool to analyze and discover concepts in Boolean datasets (i.e. Formal contexts). It does also provide a tool to analyze complex attributes by transforming them into Boolean ones (i.e. items) thanks to *conceptual scaling* (see see Section 3.2.3.2). For instance, a numerical attribute whose values are $\{1,2,3\}$ can be transformed to the set of items $\{\leq 1, \leq 2, \leq 3, \geq 3, \geq 2, \geq 1\}$ thanks to interordinal scaling. Such transformations allow us to use standard algorithms like CLOSE-BY-ONE (CBO) to look for concepts in complex datasets by leveraging a closure operator (see Algorithm 3 and Algorithm 4). However, these standard algorithms do not use the relationships between items to enumerate the concepts as for example the fact that ≤ 1 implies ≤ 2 and so on. For such, they can perform additional closure computations which substantially degrade their performance. We propose in this section a generic algorithm, named CBOI for CLOSE-BY-ONE USING IMPLICATIONS, to enumerate concepts (or equivalently extents) in a formal context using the inherent implications between items provided as an input. We show that using the implications between items can reduce significantly the number of closure computations and hence the time effort spent to enumerate the whole set of concepts. This contribution appeared in paper [16] on which the writing of this section rely.

As the proposed algorithm CBOI relies on the notion of **item-implications**, we will recall below the definition of this notion.

Definition 5.1 (Item-Implications). Let $A, B \subseteq \mathcal{M}$ be two itemsets. We say that A implies B and we denote $A \rightarrow B$ iff: $ext(A) \subseteq ext(B)$. In other words, if an object has all items in the set of attributes A then it has all items in itemset B. For two items $a, b \in \mathcal{M}$, we call an implication $\{a\} \rightarrow \{b\}$ **item-implication** and we denote it $a \rightarrow b$ for ease of notation.

Example 5.1. Consider the dataset depicted in Fig. 5.1 **(left)** and its associated *interordinal scaling* **(right)** (see Section 3.2.3.2). One should notice that inherently, we have the following item-implications $x \geq 2 \rightarrow x \geq 1$, $x \geq 3 \rightarrow x \geq 2$ and so on. By, *inherently*, we mean we do have logically $x \geq 2 \rightarrow x \geq 1$ independently from the scaled context. In fact, these implications are item-implications existing in the (interordinal) **scale** (see Definition 3.11). □

5.2.1 Problem Statement

In order to understand how *item-implications* can be used to enumerate the extents with less effort comparing to Algorithm 4, let us consider the following execution of Algorithm 4 when the formal context depicted in the right hand side of Fig. 5.1 is considered:

1. **Begins:** $C_0 = int(\mathcal{G}) = \{x \geq 1, x \leq 3, y \geq 2, y \leq 4\}$ and $B = \emptyset$ (Line 9).
2. **Add item $\mathbf{x \geq 2}$ to C_0:** $C_{new} = \{x \geq 1, \mathbf{x \geq 2}, x \leq 3, y \geq 2, y \leq 4\}$ at Line 4. The canonicity test does not fail since $B = \emptyset$ and algorithm continues by enumerating all subconcepts of $(ext(C_{new}), C_{new})$. Further, at line 7, we have $B = \{\mathbf{x \geq 2}\}$.

$\mathcal{G}$	x	y
g_1	1	4
g_2	2	2
g_3	2	2
g_4	3	2

$\mathcal{G}$	$x \geq 1$	$x \geq 2$	$x \geq 3$	$x \leq 3$	$x \leq 2$	$x \leq 1$	$y \geq 2$	$y \geq 4$	$y \leq 4$	$y \leq 2$
g_1	×			×	×	×	×	×	×	
g_2	×	×		×	×		×		×	×
g_3	×	×		×	×		×	×	×	
g_4	×	×	×	×			×	×	×	

Figure 5.1: (**left**) A numerical dataset with 2 numerical attributes. **(right)** A formal context that is the result of an *interordinal scaling* of the numerical dataset.

3. **Add item $\mathbf{x \geq 3}$ to C_0:** $C_{new} = \{x \geq 1, \underline{x \geq 2}, \mathbf{x \geq 3}, x \leq 3, y \geq 2, y \geq 4, y \leq 4\}$ at Line 4. Since $B = \{\underline{x \geq 2}\}$, canonicity test fails since $C_{new} \cap B = \{\underline{x \geq 2}\}$. The enumeration continues.

The problem shown beforehand after adding $x \geq 3$ is the fact that there was a useless closure computation that led to a certain failure. One could avoid this closure computation if the inherent implication $x \geq 3 \rightarrow x \geq 2$ is used. Indeed, since $x \geq 3 \rightarrow x \geq 2$, any closed itemset containing $x \geq 3$ contains $x \geq 2$. This shows that one can avoid some closure computations if implications are used properly. Moreover, such implications are sometime known from the user since they are *inherent* to the attributes and not derived from the incidence relation of the context. This is the case of interordinal and ordinal scaled datasets for example (see Section 3.2.3). While some state-of-the-art algorithms try to use this knowledge (i.e. implications between some items of the context) in some particular datasets as it is the case of numerical datasets (interordinal scaled contexts) [104] or datasets augmented with a taxonomy of items (ordinal scaled contexts) [17, 46]; no general algorithm has been proposed to enumerate concepts in a context while taking benefit from an arbitrary provided set of item-implications (cf. Definition 5.1).

5.2.2 From Item-Implications to an Enumeration Algorithm

Based on the beforehand observation, we present here a new algorithm dubbed CLOSE-BY-ONE USING IMPLICATIONS or CBOI for short. This algorithm has for aim to solve Problem 5.2 using the additional knowledge of implications between items.

5.2.2.1 User inputs

Definition 5.2. The **item-implication basis** of a context $\mathbb{P}$ is denoted $\rightarrow$ and is given by:

$$\rightarrow := \{(a,b) \in \mathcal{M} \times \mathcal{M} \mid a \rightarrow b\} = \{(a,b) \in \mathcal{M} \times \mathcal{M} \mid ext(a) \subseteq ext(b)\}$$

The definition given beforehand regroups all item-implications existing in the context. Moreover, $\rightarrow$ induces a pre-order on $\mathcal{M}$, that is a reflexive and transitive binary relation.

We model now the item-implication basis known/provided by the user. We can say informally that such a set of implications are those that are inherent to the attributes (not necessarily derived) from the incidence relation.

Definition 5.3. A *valid Item-Implication basis* for $\mathbb{P}$ is any *sub relation* $\mathfrak{I}$ of $\rightarrow$.

The **user input** is then a pair $(\mathbb{P}, \mathfrak{I})$ where $\mathbb{P}$ is a finite formal context and $\mathfrak{I}$ is a valid item-implication basis for $\mathbb{P}$. Figure 5.2 (left) depicts an example of the input pair $(\mathbb{P}, \mathfrak{I})$. It is clear that $\mathfrak{I}$ provides a partial information about a pre-order. Relation $\mathfrak{I}$ can be augmented to a sub pre-order $\rightarrow_{\mathfrak{I}}$ of $\rightarrow$ thanks to *reflexive closure* (i.e. adding (m, m) to $\mathfrak{I}$ for all $m \in \mathcal{M}$) and *transitive closure* (i.e. the smallest transitive relation containing $\mathfrak{I}$). Thus we will be dealing from now on with an equivalent pair $(\mathbb{P}, \rightarrow_{\mathfrak{I}})$ where $(\mathcal{M}, \rightarrow_{\mathfrak{I}})$ is a pre-ordered set.

5.2.2.2 Building a partial order

Pre-order $\rightarrow_{\mathfrak{I}}$ could contain some cycles (i.e. not anti-symmetric). One can define an equivalence relation $\leftrightarrow_{\mathfrak{I}}$ on $\mathcal{M}$ such that $(\forall a, b \in \mathcal{M})\, a \leftrightarrow_{\mathfrak{I}} b$ iff $a \rightarrow_{\mathfrak{I}} b$ and $b \rightarrow_{\mathfrak{I}} a$. Please note that if $a \leftrightarrow_{\mathfrak{I}} b$ then $ext(a) = ext(b)$. With $\mathcal{M}' = \mathcal{M}/\leftrightarrow_{\mathfrak{I}}$ the quotient set of $\mathcal{M}$ on $\leftrightarrow_{\mathfrak{I}}$ and the following relation $\leq_{\mathfrak{I}}$:

$$(\forall S_1, S_2 \in \mathcal{M}')\, S_1 \leq_{\mathfrak{I}} S_2 \text{ iff } (\exists a \in S_1, \exists b \in S_2)\, a \rightarrow_{\mathfrak{I}} b$$

One can show that $(\mathcal{M}', \leq_{\mathfrak{I}})$ does form a partially ordered set called the **quotient poset** (see Section 2.2.2). Accordingly, context $\mathbb{P} = (\mathcal{G}, \mathcal{M}, I)$ is also transformed to $\mathbb{P}' = (\mathcal{G}, \mathcal{M}', I')$ where:

$$(\forall g \in \mathcal{G}, \forall S \in \mathcal{M}') \qquad g\, I'\, S \textbf{ iff } (\forall m \in S)\, g\, I\, m$$

Figure 5.2 gives an example of such a transformation from $\mathbb{P}$ to $\mathbb{P}'$ and from $(\mathcal{M}, \rightarrow_{\mathfrak{I}})$ to $(\mathcal{M}', \leq_{\mathfrak{I}})$. Note that this transformation is a *partial column-clarification* (see Definition 3.10) in the sense that it does concern only the item-implication basis $\rightarrow_{\mathfrak{I}}$ provided by the user since we want to use only the *user inputs*. If the total item-implication basis $\rightarrow$ associated with the context is used, the beforehand transformation will be equivalent to a column clarification. Proposition 5.1 shows that looking for extents in $\mathbb{P}$ is equivalent to look for extents in the partially clarified context $\mathbb{P}'$. As such, from now on, we will consider the pair $(\mathbb{P}, \leq_{\mathfrak{I}})$ such that $(\mathcal{M}, \leq_{\mathfrak{I}})$ is a partial order. If not so, the context and the item-implication basis are transformed as shown beforehand.

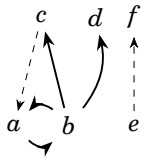
Proposition 5.1. We have $\mathbb{P}_{ext} = \mathbb{P}'_{ext}$.

Proof. Recall that $\mathbb{P}_{ext}$ and $\mathbb{P}'_{ext}$ are closed under arbitrary intersection, i.e. Moore families.

- Proof of $\mathbb{P}'_{ext} \subseteq \mathbb{P}_{ext}$: Let $A \in \mathbb{P}'_{ext}$, that is $\exists \mathbb{S} \subseteq \mathcal{M}'$ s.t. $ext_{\mathbb{P}'}(\mathbb{S}) = A$, that is: $A = \bigcap_{S \in \mathbb{S}} ext_{\mathbb{P}'}(\{S\})$. Moreover, we have: $ext_{\mathbb{P}'}(\{S\}) = \{g \in \mathcal{G} \mid g\, I'\, S\} = \{g \in \mathcal{G} \mid (\forall m \in S) g\, I\, m\} = ext_{\mathbb{P}}(S)$. Hence, $A = \bigcap_{S \in \mathbb{S}} ext_{\mathbb{P}}(S)$. Therefore, $A \in \mathbb{P}_{ext}$ since $\mathbb{P}_{ext}$ is a Moore family.
- Proof of $\mathbb{P}_{ext} \subseteq \mathbb{P}'_{ext}$: Let $A \in \mathbb{P}_{ext}$, that is: $\exists B \subseteq \mathcal{M}$ s.t. $ext_{\mathbb{P}}(B) = A$, that is: $A = \bigcap_{m \in B} ext_{\mathbb{P}}(\{m\})$. For $m \in B$, let $S_m \in \mathcal{M}'$ be the unique set containing m. We have $ext_{\mathbb{P}'}(\{S_m\}) = ext_{\mathbb{P}}(\{m\})$. Hence, $A = \bigcap_{m \in B} ext_{\mathbb{P}'}(\{S_m\})$; that is $A \in \mathbb{P}'_{ext}$ since $\mathbb{P}'_{ext}$ is a Moore family.

This conclude the proof.

$\mathcal{G}$	a	b	c	d	e	f
g_1	×	×	×	×		
g_2	×	×	×	×	×	
g_3					×	×
g_4				×		



$\mathcal{G}$	$\{\mathbf{a},\mathbf{b}\}$	$\{c\}$	$\{d\}$	$\{e\}$	$\{f\}$
g_1	×	×	×		
g_2	×	×	×	×	
g_3				×	×
g_4			×		

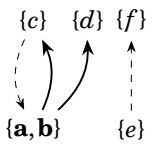


Figure 5.2: (**left**) Input context $\mathbb{P} = (\mathcal{G}, \mathcal{M}, I)$, the input-item implications $\mathfrak{I}$ (continuous arrows) and the item-implications of the context that are valid but not provided (dashed arrows). **(right)** The result of transformation to the context $\mathbb{P}' = (\mathcal{G}, \mathcal{M}', I')$ and the Hasse diagram of the poset $(\mathcal{M}', \leq_{\mathfrak{I}})$ (continuous arrows).

5.2.2.3 Closed patterns are upper ideals

Consider now the obtained pair $(\mathbb{P}, \leq_{\mathfrak{I}})$ where $\mathbb{P} = (\mathcal{G}, \mathcal{M}, I)$ is a context and $(\mathcal{M}, \leq_{\mathfrak{I}})$ is a poset s.t. $\forall a, b \in \mathcal{M}$, we have if $a \leq_{\mathfrak{I}} b$ then $ext(a) \subseteq ext(b)$.

Lemma 5.1. $\forall C \subseteq \mathcal{M} : \uparrow C \subseteq int \circ ext(C)$

Proof. Let $m \in \uparrow C$, thus $\exists c \in C$ such that $c \rightarrow_{\mathfrak{I}} m$ or in other words, $ext(c) \subseteq ext(m)$. Since int is order-reversing ((ext, int) is a Galois connection (see Proposition 3.1)), $int \circ ext(m) \subseteq int \circ ext(c)$. Since $int \circ ext$ is monotonous then $int \circ ext(c) \subseteq int \circ ext(C)$. Hence, $int \circ ext(m) \subseteq int \circ ext(C)$. By extensivity of $int \circ ext$ we conclude that $m \in int \circ ext(C)$; that is $\uparrow C \subseteq int \circ ext(C)$. ∎

A straightforward corollary of Lemma 5.1 is given in Proposition 5.2 below.

Proposition 5.2. Closed patterns are upper ideals on $(\mathcal{M}, \leq_{\mathfrak{I}})$ that is $\mathbb{P}_{int} \subseteq \mathscr{U}(\mathcal{M})$ where $\mathscr{U}(\mathcal{M}) = \{S \subseteq \mathcal{M} \mid \uparrow S = S\}$ is the set of upper ideals on $(\mathcal{M}, \leq_{\mathfrak{I}})$.

Proof. Let $C \in \mathbb{P}_{int}$, we have $C \subseteq \uparrow C$ by extensivity of $\uparrow$. In the other hand, we have $\uparrow C \subseteq int \circ ext(C) = C$ according to Proposition 5.1 and by using the impotence of $int \circ ext$. Hence, $C = \uparrow C$ or in other words $C \in \mathscr{U}(\mathcal{M})$. ∎

Note 5.2. It should be noticed that in a *column-clarified context* $\mathbb{P} = (\mathcal{G}, \mathcal{M}, I)$, the relation $\rightarrow$ (see Definition 5.2) induces a partial order on $\mathcal{M}$. Moreover, we have $(\forall m \in \mathcal{M})\ int \circ ext(m) = \uparrow m$, that is principal filters in poset $(\mathcal{M}, \rightarrow)$ are closed. □

5.2.2.4 Algorithm CLOSE-BY-ONE USING IMPLICATIONS

We have shown in Proposition 5.2 that, for a $(\mathbb{P}, \leq_{\mathfrak{I}})$ with $\mathbb{P} = (\mathcal{G}, \mathcal{M}, I)$ s.t. $(\mathcal{M}, \leq_{\mathfrak{I}})$ is a poset, all closed patterns in $\mathbb{P}_{int}$ are upper ideals in $\mathscr{U}(\mathcal{M})$. Since $(\mathcal{M}, \mathscr{U}(\mathcal{M}))$ is strongly accessible (see Proposition 4.8), we can use Algorithm 1 (D&C) to enumerate concepts in $\mathbb{P}$ using closure operator

Algorithm 5: Algorithm CLOSE-BY-ONE USING IMPLICATIONS (CBOI)

Input: Pair $(\mathbb{P}, \leq_{\mathfrak{I}})$ with $\mathbb{P} = (\mathcal{G}, \mathcal{M}, I)$ a formal context and $(\mathcal{M}, \leq_{\mathfrak{I}})$ is a poset s.t. if $m_1 \leq_{\mathfrak{I}} m_2$ then $ext(m_1) \subseteq ext(m_2)$.
Output: Elements of $\mathfrak{B}(\mathbb{P})$

```
1 procedure CbOI(A, C, B)
2     for m ∈ addables(C)\B do
3         A_new ← A ∩ ext(m)                    // Compute the new extent A_new
4         C_new ← int(A_new)                    // Compute the new intent C_new
5         if C_new ∩ B = ∅ then
6             CbOI(A_new, C_new, B)
7         B ← B ∪ {m}                  // Update the set of banned attributes B
8     Print(A)                                              // Output extent A
9 CbOI(G, int(G), ∅)
```

$int \circ ext$ on $(\mathcal{M}, \mathscr{U}(\mathcal{M}))$ rather than on $(\mathcal{M}, \wp(\mathcal{M}))$ as Algorithm 4 does. Algorithm 5 dubbed CLOSE-BY-ONE USING IMPLICATIONS *(*CBOI *for short)* is then a straightforward implementation of Algorithm 4 where only line 2 is modified according to Lemma 4.1, i.e. For $C \in \mathbb{P}_{int}$, the set addables(C) denotes the set of items to add to build the next closed itemsets and is given by:

$$\texttt{addables}(C) := max(\mathcal{M} \setminus C) = \{a \in \mathcal{M} \setminus C \mid uppers(a) \subseteq C\}$$

Example 5.2. Figure 5.3 depicts for $C \in \mathscr{U}(\mathcal{M})$ the set $max(\mathcal{M} \setminus C)$. Hence, the only direct upper neighbors of $C = \{\mathbf{a}, \mathbf{c}, \mathbf{d}\}$ in $(\mathscr{U}(\mathcal{M}), \subseteq)$ according to Lemma 4.1 are $\{\mathbf{a}, \mathbf{c}, \mathbf{d}, \underline{b}\}$, $\{\mathbf{a}, \mathbf{c}, \mathbf{d}, \underline{e}\}$ and $\{\mathbf{a}, \mathbf{c}, \mathbf{d}, \underline{f}\}$ since $max(\mathcal{M} \setminus C) = \{\underline{b}, \underline{e}, \underline{f}\}$. □

In the next section, we show that some optimizations can be made to compute and maintain efficiently the set of addable items for each generated closed sets (Line 2). Moreover, one can partially compute the closure (Line 3-4) in order to perform canonicity test (Line 5).

5.2.3 Empirical Evaluation and Technical Details

We start by explaining some technical details around CBOI provided in the following link :

`https://github.com/BelfodilAimene/CbOImplications`

5.2.3.1 Implementation details

Computing the Partial Order from the Input Item-Implications. As explained in Section 5.2.2.2, the user input is a pair $(\mathbb{P}^{(0)}, \mathfrak{I}^{(0)})$ where context $\mathbb{P}^{(0)} = (\mathcal{G}, \mathcal{M}^{(0)}, I^{(0)})$ a finite context and $\mathfrak{I} \subseteq \mathcal{M}^{(0)} \times \mathcal{M}^{(0)}$ such that for all $(m_1, m_2) \in \mathfrak{I}$ we have $ext(m_1) \subseteq ext(m_2)$ (i.e. valid item-implication basis). One can model this provided item-implication basis as a directed graph $(\mathcal{M}^{(0)}, \mathfrak{I}^{(0)})$. The aim at the beginning, is to compute the associated partial order and the partial

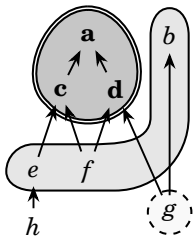


Figure 5.3: Poset $(\mathcal{M}, \leq_{\mathfrak{I}})$ with $\mathcal{M} = \{a, b, c, d, e, f, g, h\}$. The set *contoured with two lines* $C = \{\mathbf{a}, \mathbf{c}, \mathbf{d}\} \in \mathscr{U}(\mathcal{M})$ is an upset and $min(C) = \{c, d\}$. The set *contoured with one line* $\texttt{addables}(C) = max(\mathcal{M} \setminus C) = \{b, e, f\}$ regroups addable items. Indeed, all upper neighbors of items b, e and f are in C (note that item b has no upper-neighbor). The set *contoured with one dashed line* $\texttt{potential\_addables}(C) = \{g\}$ contains potential addable items. Indeed, $uppers(g) = \{b, \mathbf{d}\}$ contains at least one element in C.

scaling of the context w.r.t. to the item implication basis. To do so we start by computing the set of strongly connected components $\mathcal{M}$ on the directed graph $(\mathcal{M}^{(0)}, \mathfrak{I}^{(0)})$. This can be done for instance using Tarjan's Algorithm [155] whose complexity is $O(|\mathcal{M}^{(0)}| + |\mathfrak{I}^{(0)}|)$. Once done, we can build the associated *Directed Acyclic Graph (DAG)* $(\mathcal{M}, \mathfrak{I}^{(1)})$ where there is an arc (S_1, S_2) in $\mathfrak{I}^{(1)}$ iff $\exists m_1 \in S_1$ and $m_2 \in S_2$ such that $(m_1, m_2) \in \mathfrak{I}^{(0)}$. Such an operation is called **graph condensation**. Once $\mathcal{M}$ computed, the context $\mathbb{P}^{(0)}$ is transformed to an equivalent context $\mathbb{P} = (\mathcal{G}, \mathcal{M}, I)$ (i.e. $\mathbb{P}_{ext} = \mathbb{P}^{(0)}_{ext}$ according to Proposition 5.1). Notice that the obtained DAG $(\mathcal{M}, \mathfrak{I}^{(1)})$ represents the partial order $\leq_{\mathfrak{I}}$ (i.e. a reflexive and transitive closure of $\mathfrak{I}$ creates $\leq_{\mathfrak{I}}$). A more usual and efficient way to store $\leq_{\mathfrak{I}}$ is to compute the *transitive reduction* of $(\mathcal{M}, \mathfrak{I}^{(1)})$ [7] to obtain the Hasse Diagram of $\leq_{\mathfrak{I}}$. We obtain then $(\mathcal{M}, \mathfrak{I})$ where we store for each $m \in \mathcal{M}$ both sets of its direct lower and upper neighbors (i.e. $lowers(m)$ and $uppers(m)$) w.r.t. $(\mathcal{M}, \leq_{\mathfrak{I}})$.

Computing and Maintaining Addable Items. Now that the partial order $\leq_{\mathfrak{I}}$ is encoded by the list of lower and upper neighbors for each item $m \in \mathcal{M}$ (i.e. the Hasse Diagram). One solution is that at each step of the algorithm, for a closed itemset C, we computed the set $\texttt{addables}(C)$ using its formula $\{a \in \mathcal{M} \setminus C \mid uppers(a) \subseteq C\}$. Such a way of computation has a complexity of $O(\mathcal{G}^2)$ and hence could lessen the performances of the implementation of CbOI. To address this drawback, we propose to keep for each generated itemset $C \in \mathscr{U}(\mathcal{M})$, the set of its addable items $\texttt{addables}(C)$, the set of potential addable items $\texttt{potential\_addables}(C)$ and its minimal elements $min(C)$ (only if we want to output them). An item $p \in \mathcal{M}$ is said to be *potentially addable* if $p \in \mathcal{M} \setminus (C \cup \texttt{addables}(C))$ and it has at least one element of its direct upper neighbors $uppers(p)$ in C. Formally: $\texttt{potential\_addables}(C) = (\bigcup_{c \in min(C)} lowers(c)) \setminus \texttt{addables}(C)$.

Example 5.3. Figure 5.3 gives an example about addable and potential addable items for a closed itemset $C = \{a, c, d\}$. □

The three sets of *addable items*, *potential addable items* and *minimal items* can be maintained incrementally as follow. Given an up-set $C \in \mathscr{U}(\mathcal{M})$ and an item $a \in \texttt{addables}(C)$, the following steps are performed to compute the three sets associated to the up-set $C \cup \{a\}$:

1. $min(C \cup \{a\}) := (min(C) \setminus uppers(a)) \cup \{a\}$.
2. $\texttt{potential\_addables}(C \cup \{a\}) := \texttt{potential\_addables}(C) \cup lowers(a)$
3. Initialize addable items $\texttt{addables}(C \cup \{a\})$ by $\texttt{addables}(C) \setminus \{a\}$.

4. For each item p in the computed potential_addables($C \cup \{a\}$), if we have $uppers(p) \subseteq min(C \cup \{a\})$ then remove it from potential_addables($C \cup \{a\}$) and add it to addables($C \cup \{a\}$). This can be further optimized by maintaining for each potentially addable item the number of direct upper neighbors that are not already in C. Whenever element a is added, we subtract 1 from the values associated to elements in $lowers(a)$. Once a potentially addable element sees its value become 0, he is considered as addable item and no longer potentially addable.

Example 5.4. Going back to Figure 5.3, adding item $\underline{b}$ updates the different sets as follow:

- $C_{new} = \{\mathbf{a}, \mathbf{c}, \mathbf{d}, \underline{b}\}$,
- $min(C_{new}) = \{\mathbf{c}, \mathbf{d}, \underline{b}\}$,
- addables(C_{new}) $= \{e, f, \underline{g}\}$ and
- potential_addables(C_{new}) $= \emptyset$. □

Computing Next Closure and Performing Canonicity Test. Line 3-5 in Algorithm 5 are dedicated to closure computation of the newly generated set and checking if such a closed pattern is already generated. Some optimizations can be made here. For instance, vertical representation of the context (i.e. keeping for each item, its extent) can be held in memory in order to compute efficiently the new pattern extent (Line 3). For closure computation (Line 4) and canonicity test (Line 5), one can use the optimizations explained below:

1. We have a canonicity test fail (i.e. $int \circ ext(C \cup \{m\}) \cap B \neq \emptyset$) iff $\exists b \in B \cap$ addables(C) such that $ext(C \cup \{m\}) \subseteq ext(b)$. Hence, we do not need to compute the closure $int \circ ext(C \cup \{m\})$ to perform the canonicity test. Note that to ensure a fair comparison between CbOI and CbO-TD, this same optimization has been used for CbO-TD implementation.
2. To maintain both sets of addable and potential addable items as explained beforehand, closure computation is computed incrementally by adding item per item until there is no addable item a s.t. $ext(C \cup \{m\}) \subseteq ext(a)$.

Outputting Minimal Elements. If the item-implications in $\leq_{\mathfrak{I}}$ are well-known by the user, one should output only minimal element of a closed pattern C w.r.t. $\leq_{\mathfrak{I}}$ (i.e. $min(C)$) since C contains some redundant information [46, 104].

5.2.3.2 Empirical evaluation

Experiment Settings. Experiments were conducted in a machine with an Intel Core i7-7700HQ 2.80GHz CPU and *7.7 GiB* memory space and the implementation was done using Python 2.7.12. Table 5.1 reports the benchmark input contexts and their associated item-implications basis. *Europarl*[1] and *Yelp*[2] are datasets augmented with a taxonomy. Hence, their corresponding contexts $\mathbb{P}_1$ and $\mathbb{P}_2$ are obtained via an ordinal scaling and their associated implication basis are

[1]EPD8 (last accessed on 04 Octobre 2018): http://parltrack.euwiki.org/
[2]Yelp (last accessed on 25 April 2017): www.yelp.com/dataset/challenge

$(\mathbb{P},\mathfrak{I})$	**context** $\mathbb{P}=(\mathcal{G},\mathcal{M},I)$ Name	$\lvert\mathcal{G}\rvert$	$\lvert\mathcal{M}\rvert$	$\lvert\mathbb{P}_{int}\rvert$	**implications** $\mathfrak{I}$ $\lvert\rightarrow^*_{\mathfrak{I}}\rvert$	$\lvert\rightarrow^*\rvert$	density
$(\mathbb{P}_1,\mathfrak{I}_1)$	Europarl	4 742	357	1 307	709	1 034	68.57%
$(\mathbb{P}_2,\mathfrak{I}_2)$	Yelp	127 162	1 174	63 300	1 514	2 111	71.72%
$(\mathbb{P}_3,\mathfrak{I}_3)$	Basketball	40	272	272 223	4 716	13 724	34.36%
$(\mathbb{P}_4,\mathfrak{I}_4)$	Iris	150	246	6 516 292	3 964	11 704	33.87%
$(\mathbb{P}_5,\mathfrak{I}_5)$	Airport	135	1 348	82 467 125	90 182	313 432	28.77%
$(\mathbb{P}_6,\mathfrak{I}_6)$	Mushrooms	8 124	119	238 710	0	949	0.00%

Table 5.1: Benchmark Inputs and their characteristics: the number of objects $|\mathcal{G}|$, the number of attributes $|\mathcal{M}|$, the size of the concept lattice $|\mathbb{P}_{int}|$ of the context $\mathbb{P}$, the number of strict (irreflexive) item-implications $|\rightarrow^*_{\mathfrak{I}}|$ in the pre-order associated to the corresponding input implication basis, the number of strict item-implications in the context $|\rightarrow^*|$ (see Definition 5.2) and the density given by $|\rightarrow^*_{\mathfrak{I}}|/|\rightarrow^*|$.

derived from the hierarchy of items induced by the provided taxonomy. *Basketball*[3], *Airport*[3] and *Iris*[4] are numerical datasets. Analogously, their corresponding contexts $\mathbb{P}_3$, $\mathbb{P}_4$ and $\mathbb{P}_5$ are the result of an interordinal scaling and the associated implication basis are constituted with two chains of implications per attribute (i.e. if the domain of the numerical attribute is $\{1,2,3\}$ then the item implications basis associated to the inter-ordinal scaling is given by $\leq 1 \rightarrow \leq 2 \rightarrow \leq 3$ and $\geq 3 \rightarrow \geq 2 \rightarrow \geq 1$). *Mushroom*[4] features only nominal attributes. Hence, its associated context $\mathbb{P}_6$ represents the result of nominal scaling of all attributes. Note that the set of implications $\mathfrak{I}_6$ is empty, yet there are some implications between items that are context-dependent and obviously not inherent to the attributes (i.e. $\rightarrow$ is not empty).

Evaluation Results. Table 5.2 reports the number of closures and the performance of CbO-TD and CbOI on the different benchmark inputs. For each benchmark context $\mathbb{P}_i$, we run both algorithms on the provided implication basis (i.e. input $(\mathbb{P}_i,\mathfrak{I}_i)$) as well as on the total one that is associated to the context (i.e. $(\mathbb{P}_i,\rightarrow)$). For a fair comparison, we report the context load/preparation time into the memory for CbO-TD as well as the load/preparation time of the pair (context, implication basis) for CbOI. When $\rightarrow$ implication basis is used, the load time includes the time spent to compute it.

It is clear that the number of closures performed by CbOI is much less than the ones performed by CbO-TD in all tests excepts when no implications are provided. This corresponds to the case $(\mathbb{P}_6,\mathfrak{I}_6)$ where the number of closures performed by CbOI is supposed to be equivalent to the number of closures performed by CbO-TD if the same order of choice of items to add is followed.

Concerning the execution time, it is clear that the load time for CbO-TD is lesser than the load/compute time for CbOI since CbOI does load and prepare additionally the item-implication basis. However, even if CbOI has this drawback, one can see that the enumeration time is much

[3]Bilkent repository: `http://funapp.cs.bilkent.edu.tr/`

[4]UCI repository: `https://archive.ics.uci.edu/ml/index.php`

$(\mathbb{P},\mathfrak{I})$	CbO-td				CbOI			
	nb closure	load (ms)	enum (ms)	total (ms)	nb closure	load (ms)	enum (ms)	total (ms)
$(\mathbb{P}_1,\mathfrak{I}_1)$	185 418	**86**	184	270	**17 020**	220	**48**	**268**
$(\mathbb{P}_1,\rightarrow)$					**13 409**	191	**46**	**237**
$(\mathbb{P}_2,\mathfrak{I}_2)$	24 437 659	**8 715**	30 360	39 075	**3 317 590**	18 084	**16 107**	**34 191**
$(\mathbb{P}_2,\rightarrow)$					**2 974 130**	19 976	**15 030**	**35 006**
$(\mathbb{P}_3,\mathfrak{I}_3)$	13 340 233	**3**	57 286	57 289	**703 999**	20	**3 628**	**3 648**
$(\mathbb{P}_3,\rightarrow)$					**445 735**	39	**9 114**	**9 153**
$(\mathbb{P}_4,\mathfrak{I}_4)$	170 615 166	**10**	709 517	709 527	**9 618 493**	26	**77 586**	**77 612**
$(\mathbb{P}_4,\rightarrow)$					**8 383 741**	73	**141 016**	**141 089**
$(\mathbb{P}_5,\mathfrak{I}_5)$	NA	**53**	$> 12h$	$> 12h$	**122 717 962**	268	**1 496 175**	**1 496 443**
$(\mathbb{P}_5,\rightarrow)$					**106 409 230**	1 400	**8 221 648**	**8 223 048**
$(\mathbb{P}_6,\mathfrak{I}_6)$	**4 363 487**	**155**	**13 800**	**13 955**	4 363 511	184	15 985	16 169
$(\mathbb{P}_6,\rightarrow)$	4 363 487	**155**	13 800	13 955	**1 338 245**	244	**10 003**	**10 247**

Table 5.2: CbO-td and CbOI performance comparison on the benchmark inputs

faster than CbO-td (i.e. up to 15× faster for input $(\mathbb{P}_3,\mathfrak{I}_3)$ or even more for input $(\mathbb{P}_4,\mathfrak{I}_4)$). This compensates the overhead induced by the implication-basis load time in CbOI. One could notice that CbO-td performs better than CbOI when no implication is provided as it is the case in test $(\mathbb{P}_6,\mathfrak{I}_6)$. This is due to the fact that CbOI manages more structures than CbO-td during enumeration. It is worth noting that even if the implication basis is computed then used to enumerate concepts (see tests $(\mathbb{P}_i,\rightarrow)$), CbOI performs faster than CbO-td (up to 6× faster for input $(\mathbb{P}_3,\mathfrak{I}_3)$). Still, we can observe that CbOI is less efficient when the underlying implication basis is huge (case $(\mathbb{P}_3,\rightarrow)$, $(\mathbb{P}_4,\rightarrow)$ and $(\mathbb{P}_5,\rightarrow)$). This can be explained by the fact that CbOI spends more time to handle a huge and complex system of item-implications but the gain obtained from these base of implications does not compensate this effort.

5.2.4 Conclusion

In this section, we have investigated how to incorporate and leverage the inherent implications between items in some given context so as to enumerate more efficiently its extents. Experimental studies demonstrated that algorithm CbOI for Close-by-One using Implications is far more efficient than its concurrent CbO-td in most configurations even if the implication basis between attributes are computed in the pre-processing phase. Indeed, many aspects of the devised algorithm can be considerably improved. For instance, including FCbO optimizations [107, 136] during the enumeration process can significantly reduce the number of falsely generated closed patterns, i.e. canoncity test fails. Moreover, the load/preparation time of CbOI can be more enhanced by, for example, computing more efficiently the transitive reduction of the implication basis [116].

5.3 Enumeration in Pattern Structures

In Section 5.1 and Section 5.2, we have investigated the problem of enumerating extents when a formal context is considered. We have seen also that, thanks to conceptual scaling, the proposed algorithms can be easily used on other kind of datasets/patterns. As it is more natural to consider pattern languages directly as ordered set, we will investigate in this section the following Problem:

Problem 5.3. Let $\mathbb{P} = (\mathcal{G}, (\mathcal{D}, \sqsubseteq), \delta)$ be a finite pattern structure (i.e. set of objects $\mathcal{G}$ is finite). List **all** definable sets in $\mathbb{P}_{ext}$ **non-redundantly**.

Note 5.3. All the notations used here follow Section 3.4 on pattern structures. □

Problem 5.3 is again a subproblem of Problem 5.1 where $\mathbb{P}_{ext}$ is a closed under intersection set system which associated closure operator is $ext \circ int$. Hence, Algorithm 3 can directly be used to solve Problem 5.3. Indeed, the only difference is to replace the intersection in Line 3 by the meet $\sqcap$ operator. Algorithm 6 rewrites Algorithm 3 for pattern structures which follow the same spirit of CbO proposed in [109, 110].

Theorem 5.3. Algorithm 6 solves Problem 5.3 with:

$$\begin{aligned} &\textit{delay } O\big(\big(T_{\sqcap} + |\mathcal{G}| \cdot T_{\sqsubseteq}\big) \cdot |\mathcal{G}|\big), \text{ and} \\ &\textit{total } O\big(\big(T_{\sqcap} + |\mathcal{G}| \cdot T_{\sqsubseteq}\big) \cdot |\mathcal{G}| \cdot \mathbb{P}_{ext}\big), \text{ and} \\ &\textit{space } O\big(S_{\sqcap} + S_{\sqsubseteq} + |\mathcal{G}| \cdot \big(|\mathcal{G}| + S_{\mathcal{D}}\big)\big) \end{aligned}$$

Where $T_{\sqcap}$ ($T_{\sqsubseteq}$) and $S_{\sqcap}$ ($S_{\sqsubseteq}$) are respectively the time and space needed to perform $\sqcap$ ($\sqsubseteq$) operation. $S_{\mathcal{D}}$ is the space needed to store the largest description in $\mathbb{P}_{int}$.

Proof. The proof follows the same idea of the proof of Theorem 4.2. To see the delay complexity, one should notice that the algorithm cannot backtrack more that $|\mathcal{G}|$ times without printing an element. Each call of CbO-PS costs one meet computation $T_{\sqcap}$, $|\mathcal{G}|$ subsomption $\sqsubseteq$ checking to compute the extent $T_{\sqsubseteq}$ and manipulation (intersection, union) of sets of size $O(|\mathcal{G}|)$. The formula of delay complexity follows. The total complexity is directly deducted from the delay complexity. The space complexity proof follows also from the fact that there is no more than $|\mathcal{G}| + 1$ nested call of CbO-PS algorithm. Notice that in each call CbO-PS manipulates a constant set of variable of size $O(|\mathcal{G}|)$ and description which representation size is below the maximum possible size $S_{\mathcal{D}}$. We add to that the space size needed to compute both $S_{\sqcap}$ and $S_{\sqsubseteq}$. The formula of space complexity follows. ▌

It is clear that Algorithm 6 solves Problem 5.3 whenever we do know how to compute the meet $\sqcap$ and how to check $\sqsubseteq$. Moreover, its complexity highly depends on the complexity of $\sqcap$ and $\sqsubseteq$. Such a task could be polynomial to the input size for the case of Interval pattern structure [104], convex polygon pattern structure [20], partition pattern structure [49] among other. However, it

Algorithm 6: Algorithm CLOSE-BY-ONE *for pattern structures* (CBO-PS)

Input: $\mathbb{P} = (\mathcal{G}, (\mathcal{D}, \sqsubseteq), \delta)$ a pattern structure
with $\sqcap$ is the meet and $\top$ is the top element of $\mathcal{D}$.
Output: Elements of $\mathbb{P}_{ext}$

```
1 procedure CbO-ps(A, d, B)
2     for g ∈ 𝒢\(B ∪ A) do
3         d_new ← d ⊓ δ(g)                          // Compute the new intent d_new
4         A_new ← ext(d_new)                        // Compute the new extent A_new
5         if A_new ∩ B = ∅ then
6             CbO-ps(A_new, d_new, B)
7         B ← B ∪ {g}                               // Update the set of banned objects B
8     Print(A)                                      // Output extent A
9 CbO-ps(ext(⊤), ⊤, ∅)
```

could also be NP-hard as it is the case for graph pattern structure [110] and sequence pattern structure [39, 48] making the algorithm not even POLY-DELAY.

Algorithm 6 enumerates element of $\mathbb{P}_{ext}$ in a *depth-first search (DFS)* and *bottom up* fashion. Hence, extents are enumerated in increasing support making the algorithm well-suited for enumerating rare patterns, i.e. patterns with low support [2, 152] but not well-fashioned for the task of enumerating frequent patterns. Creating a general algorithm for Problem 5.3 enumerating extents in a *top-down* depends on how a closed pattern need to be refined to a less permissive one, i.e. equivalent to adding one item to the closed pattern for formal contexts in Algorithm 4.

In the following of this chapter and in order to propose top-down algorithms to solve Problem 5.3, we will investigate two particular instances of pattern structures on numerical data. Namely, Interval Pattern Structure proposed in [104] and Convex Polygon Pattern Structure proposed in [20]. We will often use the term closed pattern enumeration in these both language. This task is in fact equivalent to Problem 5.3 since there is a one-to-one correspondence between the set of intents $\mathbb{P}_{int}$ and the set of extents $\mathbb{P}_{ext}$.

5.4 Enumeration in Interval Pattern Structures

Before introducing interval patterns, let us recall first what a **numerical dataset** is (see Definition 3.1 for the definition of a dataset), we define below the notion of numerical dataset.

Definition 5.4. A **numerical dataset** is a **dataset** $(\mathcal{G}, \mathcal{M})$ where attributes $m \in \mathcal{M}$ has for range of value $\mathbb{R}$, i.e. $m : \mathcal{G} \to \mathbb{R}$.

As we will deal only with finite numerical datasets, from now on the set of numerical attributes $\mathcal{M}$ is given by $\{m_i \mid i \in 1 \leq i \leq |\mathcal{M}|\}$. We will also see the set $\mathcal{M}$ as a mapping:

$$\mathcal{M} : \mathcal{G} \to \mathbb{R}^{|\mathcal{M}|}, g \mapsto (m_i(g))_{1 \leq i \leq |\mathcal{M}|}$$

Example 5.5. Fig. 5.4 (left) depicts a numerical dataset $(\mathcal{G}, \mathcal{M})$ with two attributes and five objects. We have for instance $\mathcal{M}(g_3) = (2,3)$. □

When dealing with numerical domains, we generally consider for intelligibility *interval patterns* [104]. An *Interval pattern* is a conjunction of restrictions over the numerical attributes; i.e. a set of conditions $attribute \gtrless v$ with $\gtrless \in \{=, \leq, <, \geq, >\}$. Geometrically, *interval patterns* are *axis-parallel hyper-rectangles*.

Example 5.6. Fig. 5.4 **(center)** depicts an interval pattern $d = (1 \leq m_1 \leq 2) \wedge (1 \leq m_2 \leq 4)$ which can be seen as a subset of $\mathbb{R}^2$ given by $[1,2] \times [1,4]$. □

Intuitively, if we reconsider pattern $[1,2] \times [1,4]$ in Fig. 5.4 **(center)**, its extent will be $\{g_1, g_2, g_3\}$. We seek now to define the *pattern structure* related to interval patterns. In order to do that we start by studying the **Interval pattern language**.

5.4.1 Interval Pattern Language

Definition 5.5. An **interval** (in $\mathbb{R}$) $I \subseteq \mathbb{R}$ is a **convex subset** of $\mathbb{R}$, that is:

$$\forall x, y \in I, \forall z \in \mathbb{R} : x \leq z \leq y \implies z \in I$$

The set of all intervals in $\mathbb{R}$ is denoted $\mathfrak{C}(\mathbb{R})$. This set can be ordered by the standard set inclusion order $\subseteq$ to from poset $(\mathfrak{C}(\mathbb{R}), \subseteq)$.

Theorem 5.4. The poset $(\mathfrak{C}(\mathbb{R}), \subseteq)$ is a closure system on the powerset lattice $(\wp(\mathbb{R}), \subseteq)$. That is, the intersection of an arbitrary set of intervals $S \subseteq \mathfrak{C}(\mathbb{R})$ is an interval.

Proof. Let $S \subseteq \mathfrak{C}(\mathbb{R})$, we need to show that $\bigcap S \in \mathfrak{C}(\mathbb{R})$. For $S = \emptyset$, it is clear that it is the case since the top element $\mathbb{R}$ is an interval by definition. Let us show now that $\bigcap S$ is an interval when $S \neq$. If $\bigcap S = \emptyset$ then it is an interval by definition. Suppose that $\bigcap S \neq \emptyset$. Let $x, y \in \bigcap S$, then $\forall I \in S$, we have $[x, y] \subseteq S$. Therefore $[x, y] \subseteq \bigcap S$.

Hence, $(\mathfrak{C}(\mathbb{R}), \subseteq)$ is a *complete lattice* where the top element is $\mathbb{R}$ and the bottom element is $\emptyset$. The meet of $S \subseteq \mathfrak{C}(\mathbb{R})$ in this complete lattice is given by $\bigcap S$ while the join is given by the smallest interval enclosing $\bigcup S$, i.e. its convex hull. Let us investigate now the space of p-dimensional hyper-rectangles.

Definition 5.6. Let $p \in \mathbb{N}^*$, an **axis-parallel p-dimensional hyper-rectangle** $d \subseteq \mathbb{R}^p$ is the result of the product of p non-empty intervals in $\mathfrak{C}(\mathbb{R})$. Formally:

$$d = \bigtimes_{k=1}^{p} I_k \quad \text{with} \quad \forall k \in \{1..p\} : I_k \in \mathfrak{C}(\mathbb{R}) \backslash \{\emptyset\}$$

By considering $\emptyset$ also an **axis-parallel p-dimensional hyper-rectangle**, the set $\mathcal{D}_p$ denotes the set of all possible axis-parallel p-dimensional hyper-rectangles in $\mathbb{R}^p$ and can be naturally ordered by $\subseteq$.

Note 5.4. Another characterization of $\mathcal{D}_p$ is that $d \subseteq \mathbb{R}^p$ is a **axis-parallel p-dimensional hyper-rectangle** iff:

$$\left(\forall (x_i)_{1 \leq i \leq p} \in d, (y_i)_{1 \leq i \leq p} \in d\right) \bigtimes_{i=1}^{p} [inf\{x_i, y_i\}, sup\{x_i, y_i\}] \subseteq d$$

□

Theorem 5.5. For any $p \in \mathbb{N}^*$, poset $(\mathcal{D}_p, \subseteq)$ is a closure system on the powerset lattice $(\wp(\mathbb{R}^p), \subseteq)$. That is, the intersection of an arbitrary set of axis-parallel p-dimensional hyper-rectangle $S \subseteq \mathcal{D}_p$ is also a p-dimensional hyper-rectangle.

Proof. Let $S \subseteq \mathcal{D}_p$, we need to show that $\bigcap S \in \mathcal{D}_p$. For $S = \emptyset$, it is clear that it is the case since the top element $\mathbb{R}^p$ is in $\mathcal{D}_p$. Let us show now that $\bigcap S$ belongs to $\mathcal{D}_p$ when $S \neq \emptyset$. If $\bigcap S = \emptyset$ then it is in $\mathcal{D}_p$ by definition. Suppose that $\bigcap S \neq \emptyset$ and let $(x_i)_{1 \leq i \leq p}, (y_i)_{1 \leq i \leq p} \in S$ then using Note 5.4 and since $\bigtimes_{i=1}^{p} [inf\{x_i, y_i\}, sup\{x_i, y_i\}] \subseteq d$ for every single description $d \in S$ then $\bigtimes_{i=1}^{p} [inf\{x_i, y_i\}, sup\{x_i, y_i\}] \subseteq \bigcap S$.

Hence, we conclude that for any $p \in \mathbb{N}^*$, poset $(\mathcal{D}_p, \subseteq)$ is a *complete lattice* which top and bottom elements are respectively $\mathbb{R}^p$ and $\emptyset$. Moreover, the meet of a subset $S \subseteq \mathcal{D}_p$ is $\bigcap S$ while the join is given by the smallest axis-parallel p-dimensional hyper-rectangle enclosing it. Particularly, for

$\mathcal{G}$	m_1	m_2
g_1	1	1
g_2	2	2
g_3	2	3
g_4	3	4
g_5	3	1

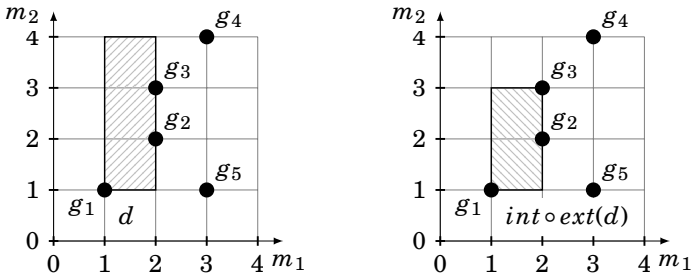


Figure 5.4: 2-dimensional numerical dataset with 5 objects **(left)**, Non-closed interval pattern d **(center)** and closed interval pattern $int \circ ext(d)$ **(right)**.

2 non-empty axis-parallel hyper-rectangles $d_j = \times_{k=1}^{p} I_k^j$ with $j \in \{1,2\}$ and $I_k^j = \left[a_k^j, b_k^j\right]$ we have:

$$d_1 \wedge d_2 = \bigtimes_{k=1}^{p} \left(I_k^1 \wedge I_k^2\right) = \bigtimes_{k=1}^{p} \left[sup\left\{a_k^1, a_k^2\right\}, inf\left\{b_k^1, b_k^2\right\}\right]$$
$$d_1 \vee d_2 = \bigtimes_{k=1}^{p} \left(I_k^1 \vee I_k^2\right) = \bigtimes_{k=1}^{p} \left[inf\left\{a_k^1, a_k^2\right\}, sup\left\{b_k^1, b_k^2\right\}\right]$$

We have seen that in pattern setups, patterns are ordered from the less restrictive to the most restrictive ones, we define the interval pattern language formally as follow:

Definition 5.7. Let $p \in \mathbb{N}^*$, the **interval pattern language** is the complete lattice $(\mathcal{D}_p, \sqsubseteq)$ where the order $\sqsubseteq$ is given by the dual inclusion $\supseteq$.

Note 5.5. It is clear $(\mathcal{D}_p, \sqsubseteq)$ is a complete lattice since it is the dual poset of a complete lattice. Hence, the meet and the join are reversed. Particularly, the meet $\sqcap$ and the join $\sqcup$ for two non-empty descriptions $d_j = \times_{k=1}^{p} I_k^j$ with $j \in \{1,2\}$ and $I_k^j = \left[a_k^j, b_k^j\right]$ we have:

$$d_1 \sqcap d_2 = \bigtimes_{k=1}^{p} \left[inf\left\{a_k^1, a_k^2\right\}, sup\left\{b_k^1, b_k^2\right\}\right]$$
$$d_1 \sqcup d_2 = \bigtimes_{k=1}^{p} \left[sup\left\{a_k^1, a_k^2\right\}, inf\left\{b_k^1, b_k^2\right\}\right]$$

The meet and the join of other types of interval patterns (i.e. semi-open and open) can be deduced accordingly. □

Example 5.7. Fig. 5.5 depicts the meet and join of two interval patterns d_1, d_2 in $(\mathcal{D}_2, \sqsubseteq)$. We have $d_1 = [1,5] \times (1,4]$, $d_2 = [0,4) \times [2,6]$. $d_1 \sqcap d_2 = [0,5] \times (1,6]$ is the smallest rectangle enclosing both d_1 and d_2 and $d_1 \sqcup d_2 = [1,4) \times [2,4]$ is the intersection of the two rectangles d_1 and d_2. □

5.4.2 Interval Pattern Structure

Now that we have seen that interval pattern language $(\mathcal{D}_p, \sqsubseteq)$ is a complete lattice. One can build a pattern structure using it according to Theorem 3.2.

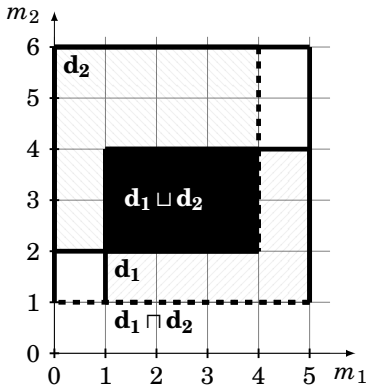


Figure 5.5: The meet and join of two interval patterns d_1, d_2 in $(\mathcal{D}_2, \sqsubseteq)$. We have $d_1 = [1,5] \times (1,4]$, $d_2 = [0,4) \times [2,6]$. $d_1 \sqcap d_2 = [0,5] \times (1,6]$ is the smallest rectangle enclosing both d_1 and d_2 and $d_1 \sqcup d_2 = [1,4) \times [2,4]$ is the intersection of the two rectangles d_1 and d_2.

Definition 5.8. Let $(\mathcal{G}, \mathcal{M})$ be a finite numerical dataset , the associated interval pattern structure is given by $\mathbb{P} = (\mathcal{G}, (\mathcal{D}, \sqsubseteq), \delta)$ with $\mathcal{D} := \mathcal{D}_{|\mathcal{M}|}$ (see Definition 5.7) and δ is given by:

$$\delta : \mathcal{G} \to \mathcal{D}, g \mapsto \{\mathcal{M}(g)\} = \bigtimes_{i=1}^{|\mathcal{M}|} [m_i(g), m_i(g)]$$

The extent and the intent can be formulated as follow:

$$ext : \mathcal{D} \to \wp(\mathcal{G}), d \mapsto \{g \in \mathcal{G} \mid \mathcal{M}(g) \in d\}$$
$$int : \wp(\mathcal{G}) \to \mathcal{D}, A \mapsto \bigtimes_{i=1}^{|\mathcal{M}|} [min(m_i[A]), max(m_i[A])]$$

with $m_i[A] = \{m_i(g) \mid g \in A\}$.

Example 5.8. Consider the numerical dataset presented in Fig. 5.4 **(left)**. For $g_3 \in \mathcal{G}$, we have $\delta(g_3) = [2,2] \times [3,3]$. Consider now the pattern $d = [1,2] \times [1,4]$ depicted in Fig. 5.4 **(center)**, it is clear that $ext(d) = \{g_1, g_2, g_3\}$. Moreover, d is not closed since $int \circ ext(d) = int(\{g_1, g_2, g_3\}) = [1,2] \times [1,3] \neq d$. The closed pattern $int \circ ext(d)$ is depicted in Fig. 5.4 **(right)**. □

Note 5.6. An equivalent formalization of the interval pattern structure is the use of the notion of the direct product of other m base pattern structure $(\mathcal{G}, (\mathfrak{C}(\mathbb{R}), \supseteq), \delta)$ (see Section 3.7.3). □

5.4.3 Closed Interval Pattern Enumeration

It is clear that someone can directly use Algorithm 6 to enumerate in a bottom-up fashion the set of all extents in $\mathbb{P}_{ext}$ since $\mathbb{P}$ is a pattern structure. We will now explore how to enumerate closed interval patterns in $\mathbb{P}_{int}$ and equivalently elements of $\mathbb{P}_{ext}$ in a top-down fashion, i.e. in decreasing support.

It is important here to notice that the interval pattern language $(\mathcal{D}, \sqsubseteq)$ considered here in this section is infinite. However, [104] considered a finite subset of this description language $(\mathcal{D}_{\mathcal{G}}, \sqsubseteq)$

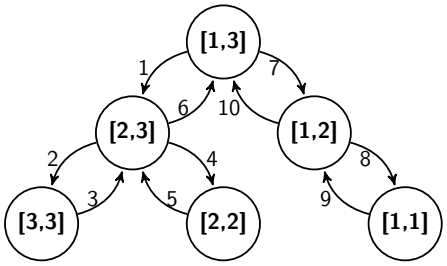


Figure 5.6: Enumerating exhaustively and non-redundantly closed intervals in $\mathbb{R}$ using only elements in $\{1,2,3\}$ in top-down depth-first fashion.

where the considered axis-parallel hyper-rectangles are built over closed intervals using only the values appearing in the dataset. In other words, $\mathcal{D}_{\mathcal{G}}$ regroups all closed rectangles fitting the grid using the different values of the dataset on each axis. It is formally given by:

$$\mathcal{D}_{\mathcal{G}} := \left\{ \bigtimes_{i=1}^{|\mathcal{M}|} [a_i, b_i] \mid (\forall i \in \{1,...,|\mathcal{M}|\})\ a_i, b_i \in m_i[\mathcal{G}] \right\}$$

Technically, $(\mathcal{D}_{\mathcal{G}}, \sqsubseteq)$ is a (complete) sublattice of $(\mathcal{D}, \sqsubseteq)$ containing all closed patterns in $\mathbb{P}_{int}$. [104] proposed an algorithm to enumerate all elements $\mathcal{D}_{\mathcal{G}}$ exhaustively and non-redundantly in a top-down fashion.

5.4.3.1 Enumerating elements of $\mathcal{D}_{\mathcal{G}}$

We start here to explain how someone can enumerate exhaustively all closed intervals built using values in $\{1,2,3\}$ in a top-down fashion. The algorithm proposed in [104] starts from the top interval $[1,3]$. Then, as shown in Figure 5.6, at every step of the algorithm two minimal changes are applied: minimal left change (`minLeftChange`) and minimal right change (`minRightChange`). To ensure a non-redundant generation, a `minLeftChange` is not allowed after `minRightChange`. Interestingly, this Algorithm can be see as an implementation of Algorithm `ExR` presented in Section 4.3.4 for enumerating convex sets in a convex geometry.

Consider now the task of enumerating all elements of $\mathcal{D}_{\mathcal{G}}$ where we have $|\mathcal{M}|$ numerical attributes. The algorithm is quite the same with two main differences to ensure non-redundancy: (1) It considers a total order on the set of attributes $\mathcal{M}$ using the index of attributes; (2) When a minimal change is applied to the attribute of index i, only attributes of index $j >= i$ can be refined in further steps from the generated pattern.

5.4.3.2 MININTCHANGE - Enumerating extents in a top-down fashion

Since all the intents of the interval pattern structure $\mathbb{P}$ are in $\mathcal{D}_{\mathcal{G}}$, Algorithm 7 [104], namely MININTCHANGE, uses the algorithm explained beforehand along with the closure operator $int \circ ext$ in the same spirit of CLOSE-BY-ONE (CBO). Algorithm 7 starts from $int(\mathcal{G})$ (Line 18) then Follows the same schema explained beforehand to enumerate elements of $\mathcal{D}_{\mathcal{G}}$. One should

Algorithm 7: Algorithm MININTCHANGE

Input: $(\mathcal{G},\mathcal{M})$ a finite numerical datasets
Output: Elements of $\mathbb{P}_{ext}$ with $\mathbb{P}$ is the interval pattern structure (see Definition 5.7)

```
1  procedure MinIntChange(A, d, i, f)
                    // extent A, intent d, last argument i and f for minChange
2      for j ∈ {i, ..., |M|} do
3          f_start ← 0 if j > i else f
4          for f_new ∈ {f_start, 1} do
5              d' ← minChange(d, j, f_new)
6              A_new ← ext(d')
7              d_new ← int(A_new)
8              if (∀k ∈ {1, ..., j − 1}) d_new.I_k = d.I_k then
9                  MinIntChange(A_new, d_new, j, f_new)
10     print(A)                                              // Output extent A
11 function minChange(d, j, f)
       // minLeftChange (minRightChange) on the j^th interval of d, i.e. d.I_j if f = 0 (f = 1)
12     d' ← d                                    // copy d = ×_{1≤i≤k} [a_k, b_k] on d'
13     if f = 0 then
14         d'.I_j ← [next(a_j, m_j[G]), b_j]      // next(a_j, m_j[G]) = inf{v ∈ m_j[G] | a_j < v}
15     else
16         d'.I_j ← [a_j, prev(b_j, m_j[G])]      // prev(b_j, m_j[G]) = sup{v ∈ m_j[G] | v < b_j}
17     return d'                                  // if d'.I_j is empty return ∅
18 MinIntChange(G, int(G), 1, 0)          // Start from the closed pattern int(G)
```

notice that after each `minLeftChange` or `minRightChange` on the pattern d (Line 5) on attribute j, the algorithm creates a new pattern d' then performs the closure $int \circ ext(d')$ to create d_{new} (Line 6-7). It performs then the canonicity test on the newly created pattern d_{new} (Line 8) to check if the closed pattern d_{new} and equivalently its extent A_{new} has been already visited. If the closed pattern d_{new} differs from d on the interval of some attribute of index k coming before the last altered interval index j then the canonicity test fail. Otherwise, the algorithm continues the enumeration starting from d_{new} (Line 9).

Note 5.7. As explained in [104], another way to enumerate closed interval patterns in a *top-down* fashion is to use an interordinal scaling then using Algorithm 4. Authors of [104] showed that such a technique is not competitive comparing to using MININTCHANGE. However, one should notice that one can use Algorithm CBOI [16] presented in Section 5.2 to take benefit from the inherent implications existing in an interordinal scaled context (i.e. $\leq a$ *implies* $\leq b$ and $\geq b$ *implies* $\geq a$ if we have $a \leq b$). In fact, Algorithm CBOI behaves almost as Algorithm MININTCHANGE does to enumerate closed interval patterns. □

5.5 Enumeration in Convex Set Patterns Structures

We have seen in Section 5.4 that when we consider numerical datasets, we deal in general with interval patterns, i.e. axis-parallel hyper-rectangles. However, the choice of of the type of patterns on a dataset should depend on the knowledge we want to extract from a given dataset. Consider for instance the following example:

Example 5.9. Let be the toy dataset depicted in Fig. 5.7 **(Left)** describing 6 representative individuals with two numerical attributes, i.e. the age and the number of cigarettes smoked per week, and one boolean attribute saying if the individual has a lung cancer or not. The representation of the dataset in the Euclidean Plane is depicted in Fig. 5.7 **(right)** where white (resp. black) dots represents individuals having (resp. not having) lung cancer.

It is easy to see that individuals having cancer, i.e. $\{g_4, g_5, g_6\}$ are not separable by an interval pattern. Indeed, the smallest interval description enclosing them is given by: "$20 \leq age \leq 70$ **and** $2 \leq nb\ cig./week \leq 14$". However and clearly, the set $\{g_4, g_5, g_6\}$ is linearly separable. For instance the linear inequality "$5 \cdot (nb\ cig./week) + age \geq 60$" depicted in Fig. 5.7 **(right)** clearly isolates only individuals having lung cancer. □

This example shows that other pattern languages than interval patterns should be considered to extract other useful information from a numerical dataset. To this aim, we propose in this section a new language on numerical datasets which is the language of **conjunctions of linear inequalities**. As a linear inequality is a halfspace and any (topologically) closed convex set can be seen as the intersection of halfspaces [35] (i.e. conjunction of linear inequalities). The language of conjunctions of linear inequalities is equivalent to the **language of (closed) convex sets**. We propose hence here to study the more general language of convex sets from pattern structure perspective. Next, we propose three algorithms to enumerate separable sets induced by this language when considering finite numerical datasets. This contribution had appeared in paper [20] on which the writing of this section partly rely.

$\mathcal{G}$	age	$nb\ cig./week$	$lung\ cancer$
g_1	20	2	no
g_2	20	6	no
g_3	40	2	no
g_4	20	14	yes
g_5	40	10	yes
g_6	70	2	yes

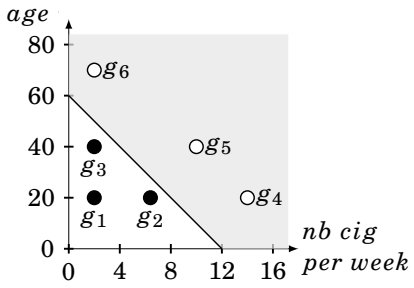


Figure 5.7: Dataset on 6 individuals **(left)** and its representation on the plane **(right)**.

5.5.1 Convex Set Pattern Language

5.5.1.1 Convex Sets

We will now start to deal with convex sets in $\mathbb{R}^p$ with $p \in \mathbb{N}^*$.

Definition 5.9. Let $a, b \in \mathbb{R}^p$, the **segment** linking a and b is denoted $\overline{ab}$ and is given by:

$$\overline{ab} := \{\lambda \cdot a + (1-\lambda) \cdot b \mid \lambda \in [0,1]\}$$

Where for $a = (a_i)_{1 \leq i \leq p}$ we have $\lambda \cdot a = (\lambda \cdot a_i)_{1 \leq i \leq p}$.

Definition 5.10. A **convex set** $S \subseteq \mathbb{R}^p$ iff $(\forall a, b \in S)$ we have $\overline{ab} \subseteq S$. The set of convex sets in $\mathbb{R}^p$ is denoted $\mathfrak{C}(\mathbb{R}^p)$.

Example 5.10. Fig. 5.8 **(left)** depicts an open disk $D = \{(x-3)^2 + (y-3)^2 < 4\}$ which center is $c = (3,3)$ and radius is 2. One can check that D is a convex set. For $a = (1,3)$ and $b = (5,3)$, all sets $D \cup \{a\}$, $D \cup \{b\}$ and $D \cup \{a,b\}$ are also convex sets in $\mathbb{R}^2$.

Fig. 5.8 **(right)** depicts a polygonal shape S which is non convex. Clearly, $a = (4, 4.5)$ and $b = (4, 1.5)$ belong to S however an element $m = (4,3)$ on the segment $\overline{ab}$ is not in S. □

Note 5.8. The convex sets in $\mathbb{R}$ are intervals (see Definition 5.5). □

Theorem 5.6. For any $p \in \mathbb{N}^*$, poset $(\mathfrak{C}(\mathbb{R}^p), \subseteq)$ is a closure system on the powerset lattice $(\wp(\mathbb{R}^p), \subseteq)$. That is, the intersection of an arbitrary set of convex sets is also a convex set.

Proof. Let $S \subseteq \mathfrak{C}(\mathbb{R}^p)$, we need to show that $\bigcap S \in \mathfrak{C}(\mathbb{R}^p)$. For $S = \emptyset$, it is clear that it is the case since the top element $\mathbb{R}^p$ is convex by definition. Let us show now that $\bigcap S$ is convex when $S \neq \emptyset$. If $\bigcap S = \emptyset$ then it is convex by definition. Suppose that $\bigcap S \neq \emptyset$. Let $a, b \in \bigcap S$, then $\forall C \in S$, we have $\overline{ab} \subseteq C$. Therefore $\overline{ab} \subseteq \bigcap S$. Hence, $\bigcap S$ is convex.

Note 5.9. Since $(\mathfrak{C}(\mathbb{R}^p), \subseteq)$ is a closure system on the powerset lattice $(\wp(\mathbb{R}^p), \subseteq)$ then its has an associated closure operator which fixpoints is $\mathfrak{C}(\mathbb{R}^p)$. This closure operator is called the **convex hull**, denoted ch, given by:

$$ch : \wp\left(\mathbb{R}^p\right) \to \wp\left(\mathbb{R}^p\right), A \mapsto \bigcap \left\{C \in \mathfrak{C}\left(\mathbb{R}^p\right) \mid A \subseteq C\right\}$$

Notice also that $(\mathfrak{C}(\mathbb{R}^p), \subseteq)$ is a complete lattice where the meet is the set intersection and the join of a set $S \subseteq \mathfrak{C}(\mathbb{R}^p)$ is given by $ch(\bigcup S)$ (see Corollary 2.1). □

Definition 5.11. Let $S \subseteq \mathfrak{C}(\mathbb{R}^p)$ be a convex set. A point $e \in S$ is said to be an **extreme point** of S iff $S \setminus \{e\}$ is convex. The set of extreme points of S is denoted $ex(S)$. A subset $B \subseteq S$ is said to be a **basis** of S iff $ch(B) = S$.

Example 5.11. In the convex set $S = D \cup \{a,b\}$ depicted in Fig. 5.8 (see Example 5.10 for the notations) we have $ex(S) = \{a,b\}$. □

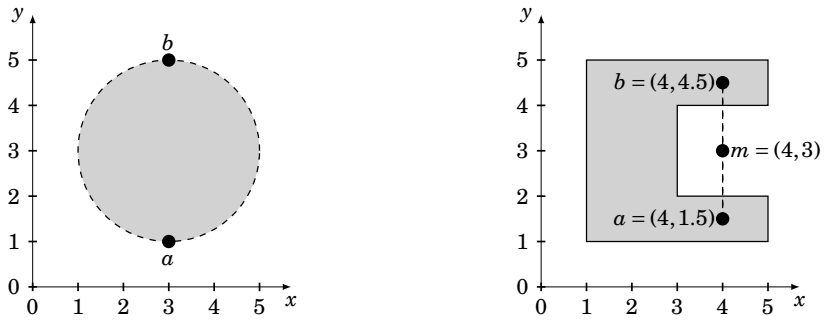


Figure 5.8: A convex set **(left)** a non convex set **(right)**.

Proposition 5.3. Let S be a convex set and let B be a **basis** of S, i.e. $ch(B) = S$. Then $ex(S) \subseteq B$.

Proof. Suppose that $\exists B \subseteq S$ s.t. $S = ch(B)$ but $\exists e \in ex(S)$ s.t. $e \notin B$. We have by definition $S \setminus \{e\}$ is convex, i.e. a fix-point of ch operator. Since $B \subseteq S \setminus \{e\}$ and since ch is order-preserving we obtain that $ch(B) \subseteq ch(S \setminus \{e\}) = S \setminus \{e\}$ which contradicts the fact that $ch(B) = S$.

5.5.1.2 Convex Polytopes

Definition 5.12. A set $S \subseteq \mathbb{R}^p$ is said to be a **convex polytope** iff $ex(S)$ is finite and it is a basis of S, i.e. $S = ch(ex(S))$. If $p = 2$, we say that S is a **convex polygon**.

Note 5.10. In fact, any set $S \subseteq \mathbb{R}^p$ having a finite basis E is a convex polytope. □

There are two ways to represent a non-empty convex polytope $S \in \mathfrak{C}(\mathbb{R}^p)$:

- The $\mathcal{V}$**-representation** or the **Vertex representation**. The convex polytope S is represented via its extreme points (i.e. the *vertices*).
- The $\mathcal{H}$**-representation** or the **Half-space representation**. The convex polytope S is seen as the solution of a system of linear inequalities $S = \{x \in \mathbb{R}^p \mid A \cdot x \leq b\}$ with $b \in \mathbb{R}^n$ and A is a matrix $n \times p$ and n represents the number of linear inequalities.

The existence of both representations is guaranteed by the *Minkowski-Weyl's Theorem* [72]. Converting one representation to another is a well known problem. The conversion from the former to the latter representation is known as the *facet enumeration problem* and the dual conversion is known as the *vertex enumeration problem*.

5.5.1.3 Convex Polygons

Particularly, when dealing with convex polygons, i.e. convex polytopes in $\mathbb{R}^2$, traversing extreme points (i.e. vertices) in a counter-clockwise order (ccw) allows to build the $\mathcal{H}$-representation from the $\mathcal{V}$-representation. To explain this observation, we present now both notions of **signed area** and **oriented segment**.

Definition 5.13. Let $a,b,c \in \mathbb{R}^2$ be three points where $a=(x_a,y_a)$, $b=(x_b,y_b)$ and $c=(x_c,y_c)$. The **signed area** of the planar triangle a,b,c (in this order) is given by:

$$\Delta(a,b,c) = \frac{1}{2} \cdot \begin{Vmatrix} x_a & y_a & 1 \\ x_b & y_b & 1 \\ x_c & y_c & 1 \end{Vmatrix} = \frac{1}{2}((y_1-y_2)\cdot x_3 + (x_2-x_1)\cdot y_3 - (x_2\cdot y_1 - x_1\cdot y_2))$$

We distinguish three cases:

- If $\Delta(a,b,c)=0$: points a,b and c are *colinear*.
- If $\Delta(a,b,c)>0$: points a,b and c form a non-empty triangle and are ordered in a counter-clockwise order.
- If $\Delta(a,b,c)<0$: points a,b and c form a non-empty triangle and are ordered in a clockwise order.

Note that $|\Delta(a,b,c)|$ gives the **area** of the triangle.

Definition 5.14. Let $a,b \in \mathbb{R}^2$ s.t. $a \neq b$, $a=(x_a,y_a)$ and $b=(x_b,y_b)$. The **oriented segment** $\overrightarrow{ab}$ subdivides $\mathbb{R}^2$ into four regions:

- **The open upper half-plane** $\overrightarrow{ab}^+$ given by the set of points $p \in \mathbb{R}$ such that abp form a non-empty triangle which vertices are ordered in counter-clockwise order, i.e. $\Delta(a,b,p)>0$. It is given by:
$$\overrightarrow{ab}^+ = \{(x,y) \in \mathbb{R}^2 \mid (y_b-y_a)\cdot x + (x_a-x_b)\cdot y < x_1\cdot y_2 - x_2\cdot y_1\}$$
- **The open lower half-plane** $\overrightarrow{ab}^-$ given by the set of points $p \in \mathbb{R}$ such that abp form a non-empty triangle which vertices are ordered in clockwise order, i.e. $\Delta(a,b,p)<0$.
$$\overrightarrow{ab}^- = \{(x,y) \in \mathbb{R}^2 \mid (y_a-y_b)\cdot x + (x_b-x_a)\cdot y < x_2\cdot y_1 - x_1\cdot y_2\}$$
For any $p \in \overrightarrow{ab}^-$, we will say that p **sees** the oriented segment $\overrightarrow{ab}$ or $\overrightarrow{ab}$ **is visible** from p.
- **The set of points on the segment** $\overline{ab}$ given by Definition 5.9.
- **The set of points that are colinear to** $\{a,b\}$ **but outside** $\overline{ab}$: It is given by the set of points $p \in \mathbb{R}^2$ s.t. $\Delta(a,b,p)=0$ but $p \notin \overline{ab}$.

Example 5.12. Consider the polygon (triangle) $c \in \mathfrak{C}(\mathbb{R}^2)$ depicted in Fig. 5.9 **(right)**. The set of its extreme points is given by $\{g_1,g_5,g_4\}$. This set is said to be the $\mathcal{V}$-representation of c, i.e. $c=ch(\{g_1,g_5,g_4\})$. As we have said beforehand, a counter-clockwise traversal of vertices of the triangles allow us to enumerate the oriented edges of the triangle $\overrightarrow{g_1g_5}$, $\overrightarrow{g_5g_4}$ and $\overrightarrow{g_4g_1}$. One can check that a point $p \in \mathbb{R}^2$ falls into triangle c iff it does not sees any of its oriented edges. This allow us for instance to build the $\mathcal{H}$-representation. For instance, the triangle c can be seen as the solution of the following linear inequations (use Definition 5.14 with $g_1=(1,1)$, $g_5=(3,1)$ and $g_4=(3,4)$):

$$\begin{pmatrix} 0 & -2 \\ 3 & 0 \\ -3 & 2 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} \leq \begin{pmatrix} -2 \\ 9 \\ -1 \end{pmatrix}$$

$\mathcal{G}$	m_1	m_2
g_1	1	1
g_2	2	2
g_3	2	3
g_4	3	4
g_5	3	1

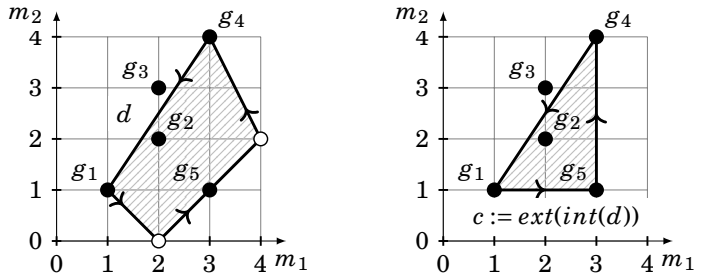


Figure 5.9: 2-dimensional numerical dataset with 5 objects **(left)**, Non-closed convex (polygon) pattern d **(center)** and closed convex (polygon) pattern $int \circ ext(d)$ **(right)**.

Equivalently, the set of points (x, y) falling into the triangle verifies the following inequalities:

$$y \geq 1 \textbf{ and } x \leq 3 \textbf{ and } 3 \cdot x - 2 \cdot y \geq 1$$

One can check indeed that $g_2 = (2,2)$ verifies the three inequations while $g_3 = (2,3)$. One can see that g_3 sees the oriented edge $\overrightarrow{g_4 g_1}$ but still verifies the two first inequations. □

Convex Polygon Representation. Regarding the last example, a convex polygon $d \in \mathfrak{C}(\mathbb{R}^2)$ will be seen as a sequence of extreme points in *counter-clockwise order* $(e_i)_{1 \leq i \leq |ex(d)|}$. This allows as presented in Example 5.12 to rebuild quickly both $\mathcal{V}$-representation $ex(d) = \{e_i\}_{1 \leq i \leq |ex(d)|}$ and its $\mathcal{H}$-representation following Example 5.12. It is clear that there is $|ex(d)|$ representations. But we will keep in mind only one representation of the polygon at a time. We denote by $|d| := |ex(d)|$ the number of extreme points of the convex polygon d, namely its **shape complexity** (i.e. it represents also the number of linear inequalities in the $\mathcal{H}$-representation). For ease of notation and when the representation $(e_i)_{1 \leq i \leq |d|}$ is fixed, we denote by $d[i] := e_i$ the i^{th} extreme points in the chosen representation.

5.5.2 Convex Set Pattern Structure

Given $p \in \mathbb{N}^*$, we have seen that poset $(\mathfrak{C}(\mathbb{R}^p), \subseteq)$ is a complete lattice (Theorem 5.6). Again, as patterns are ordered from the less restrictive (i.e. $\mathbb{R}^p$) to the most restrictive one $\emptyset$. We will consider the dual complete lattice $(\mathfrak{C}(\mathbb{R}^p), \sqsubseteq)$ where $\sqsubseteq := \supseteq$. Poset $(\mathfrak{C}(\mathbb{R}^p), \sqsubseteq)$ is called **convex set pattern language**. As the considered pattern language is a complete lattice, one can build now the pattern structure defined below.

Definition 5.15. Let $(\mathcal{G}, \mathcal{M})$ be a finite numerical dataset , one can build its associated **convex set pattern structure** $\mathbb{P} = (\mathcal{G}, (\mathcal{D}, \sqsubseteq), \delta)$ with $\mathcal{D} := \mathfrak{C}(\mathbb{R}^{|\mathcal{M}|})$ and δ is given by:

$$\delta : \mathcal{G} \to \mathcal{D}, g \mapsto \{\mathcal{M}(g)\}$$

The extent and the intent can be formulated as follow:

$$ext : \mathcal{D} \to \wp(\mathcal{G}), d \mapsto \{g \in \mathcal{G} \mid \mathcal{M}(g) \in d\}$$
$$int : \wp(\mathcal{G}) \to \mathcal{D}, A \mapsto ch(\delta[A]) = ch(\mathcal{M}[A])$$

As we are dealing with finite numerical datasets, it is clear that closed patterns are **convex polytopes** (i.e. they are the convex hull of a finite set of points). Hence, this language is equivalent the language induced by a finite conjunction of linear inequalities, in the sense that the **pattern structure of conjunctions of linear inequalities** associated to a finite numerical dataset has for set of extents $\mathbb{P}_{ext}$ (see Example 5.12).

Note 5.11. Without loss of generality, we will consider from now on a numerical dataset $(\mathcal{G}, \mathcal{M})$ where two distinct objects $g_1, g_2 \in \mathcal{G}$ s.t. $g_1 \neq g_2$ have distinct values, i.e. $\mathcal{M}(g_1) \neq \mathcal{M}(g_2)$. Interestingly, in such a case the set system $\mathbb{P}_{ext}$ form a **convex geometry** (see Definition 4.4) which associated closure operator is $ext \circ int$, called again the **convex hull**. Since objects never coincides (i.e. they are not equivalent in the sens of object implication), for ease of notation, g_i and $\mathcal{M}(g_i)$ will be used interchangeably. Notice that for $A \in \mathbb{P}_{ext}$, the objects associated to the extreme points of $int(A)$ are extreme points of A for the $ext \circ int$ closure in the convex geometry $\mathbb{P}_{ext}$. □

Example 5.13. Consider the numerical dataset presented in Fig. 5.9 **(left)**. It is clear that the convex polygon d depicted in Fig. 5.9 **(center)** is not closed w.r.t. its $ext(d) = \{g_1, g_2, g_4, g_5\}$. The closure of d is given by $c = int \circ ext(d) = ch(\delta[\{g_1, g_2, g_4, g_5\}])$ and is depicted Fig. 5.9 **(right)**. Notice that c can be represented by the sequence (g_1, g_4, g_5) in order to output easily both representations (see Example 5.12). □

5.5.3 Convex Set Pattern Structure and Interval Pattern Structure

Before presenting algorithms enumerating extents induced by the language of convex polygons, we drive the reader attention to the fact that the language of interval patterns is less expressive than the language of convex sets. Moreover, we have the following proposition:

Proposition 5.4. The poset of axis-parallel p-dimensional hyper-rectangles $(\mathcal{D}_p, \sqsubseteq)$ (see Definition 5.7) is a **kernel system** of the poset of convex sets $(\mathfrak{C}(\mathbb{R}^p), \sqsubseteq)$.

Proof. It is easy to see that $\mathcal{D}_p \subseteq \mathfrak{C}(\mathbb{R}^p)$. Moreover, recall that $(\mathcal{D}_p, \sqsubseteq)$ and $(\mathfrak{C}(\mathbb{R}^p), \sqsubseteq)$ are kernel systems (see Definition 2.14) on $(\mathbb{R}^p, \supseteq)$, i.e. they preserve arbitrary set intersection $\bigcap$ which is here the join. We conclude that $(\mathcal{D}_p, \sqsubseteq)$ is a kernel system on $(\mathfrak{C}(\mathbb{R}^p), \sqsubseteq)$. ▌

Proposition 5.4 is important since it tells also that there is a kernel operator ψ on $(\mathfrak{C}(\mathbb{R}^p), \sqsubseteq)$ which fixpoints is $\mathcal{D}_p$, i.e. ψ associates in fact the largest interval pattern w.r.t. $\sqsubseteq$ subsuming the convex set pattern (see Note 2.1). Hence, the interval pattern structure (see Definition 5.7) can be seen as a **projected pattern structure by** ψ (see Section 3.7.2) of the convex set pattern structure (see Definition 5.15) when the same numerical dataset is considered.

5.5.4 Closed Convex Pattern Enumeration

From now on we will consider only two-dimensional numerical datasets. They represents for example points described by their geo-coordinates. We will use the same notations given in Definition 5.15 and the observation given in Note 5.11, i.e. two points never collides.

5.5.4.1 ExtCbO - Using CbO-ps again

Since the convex set pattern structure $\mathbb{P}$ is a pattern structure, it is clear that someone can directly uses Algorithm 6 (CbO-ps) to enumerate exhaustively and non redundantly all elements of $\mathbb{P}_{ext}$. Again, this algorithm enumerates extents in *bottom-up fashion*. That is, it enumerates patterns in increasing support and it is well suited for rare pattern enumeration for instance.

Notice that, Algorithm ExtCbO needs to compute the convex hull of a set of points at each step, i.e. computing the intent of $A \subseteq \mathcal{G}$ (Line 3). This has for cost $\mathcal{O}(|A| \cdot log(|A|))$ when considering 2 dimensional dataset. This algorithm is clearly generalizable to any number of attributes.

5.5.4.2 DelaunayEnum - Enumerating using Delaunay triangulation

Algorithm 8, namely DelaunayEnum, enumerates closed convex polygon patterns in a Top-down fashion. This algorithm is well-fashioned for pruning search space w.r.t. minimal support, minimal perimeter, minimal area. Notice that the $area$, $permiter$ and $support$ are order-reversing mappings from $(\mathcal{D}, \sqsubseteq)$ to $\mathbb{R}$.

Algorithm 8 is in fact a direct instance of Algorithm 2, namely Extreme points Removal Closed Set Listing or ExR for short, i.e. It relies on the following procedure : to build new extents

Algorithm 8: DelaunayEnum

Input: $(\mathcal{G}, \mathcal{M})$ a finite numerical datasets with $(\forall g_i, g_j \in \mathcal{G})$ if $g_i \neq g_j$ then $\mathcal{M}(g_i) \neq \mathcal{M}(g_j)$
Output: Elements of $\mathbb{P}_{ext}$ with $\mathbb{P}$ is the convex set pattern structure (see Definition 5.15)

1 **procedure** DelaunayEnum()
2 $dt \leftarrow$ **DT**($\mathcal{G}$)
3 $d \leftarrow$ extreme points in ccw order from dt
4 **enumerate**($\mathcal{G}$,d,dt,1)
5 **procedure** enumerate (A,d,dt,pos)
6 **Print**(A) // Output extent A
7 **for** $i \in pos,...,|d|$ **do**
8 $dt_{new} \leftarrow$ **Delaunay_remove**(dt,$d[i]$) // use Algorithm in [55]
9 $d_{new} \leftarrow$ extreme points in ccw order from dt_{new} by replacing only the i^{th} extreme point of d with the new ones
10 $A_{new} \leftarrow A \backslash d[i]$ // More generally remove all objects g s.t. $\mathcal{M}(g) = d[i]$
11 **enumerate**(A_{new},d_{new}, dt_{new}, i)

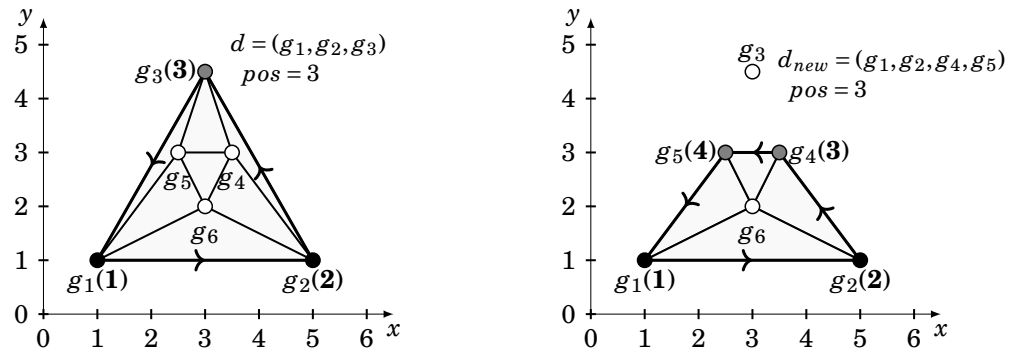


Figure 5.10: DELAUNAYENUM extreme point deletion.

from an old one $A \in \mathbb{P}_{ext}$ since $A \setminus \{e\}$ iff $e \in ex(A)$ (i.e. extreme point w.r.t $ext \circ int$ closure).

However, after deleting one extreme point, someone needs to recompute the new extreme points of the new extent (Line 3 in Algorithm 2). The easy, but slow, way, is to recompute the convex hull. However, for more a efficient update of the set of extreme points after one extreme point removal, DELAUNAYENUM uses **Delaunay triangulation (DT)** [170]. The **DT** builds in fact a mesh (graph) between objects/points in A, after a deletion of an extreme point $g_i \in ex(A)$, only its neighbors in the **DT** could become extreme points. Hence, one should maintain the **DT** after each extreme point deletion. This can be done using the algorithm proposed in [55] which time complexity is $O(k \cdot log(k))$ where k is the number of neighbors of the deleted extreme point.

Algorithm 8 enumerates *Delaunay triangulations* and maintains at each step the sequence of extreme points in counter-clockwise order, i.e. one representation. It starts by computing the DT of $\mathcal{G}$ (line 2) and then obtain the polygon representation d of $\mathcal{G}$ (line 3). At a step in the enumeration where d is the current representation, dt the current triangulation and A the current extent of d. We remove successively extreme points $p \in ex(d)$ and we update the Delaunay triangulation dt_{new} based on dt (line 8), the new representation d_{new} (sequence of extreme points) (line 9) and the new extent A_{new} by removing objects with value p from A (line 10). To avoid redundancy (avoid visiting the same pattern twice), removal of extreme points before the last removed extreme point $d[pos]$ are not allowed (argument pos in Algorithm 8) in further steps, i.e. equivalent to Line 6 in Algorithm 2. In order to do that simply, when the extreme point sequence d is updated upon extreme point $d[i]$ removal, we only insert the new extreme points (extreme points not in d) at the position i while keeping the same order (*counterclockwise*).

Example 5.14. Fig. 5.10 (from **left** to **right**) shows an example of how the algorithm updates the description d whose representation is (g_1, g_2, g_3) upon removal of the extreme point g_3 (position 3) and produces the description d_{new} whose representation is $(g_1, g_2, \underline{g_4}, \underline{g_5})$ where $\underline{g_4}$ and $\underline{g_5}$ denote the new extreme points that replaced g_3. The enumeration continues by removing $\underline{g_4}$. The edges in the both patterns represents the Delaunay Triangulation graph between points. □

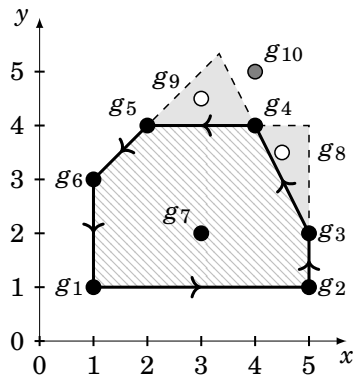


Figure 5.11: Convex polygon and candidate points

5.5.4.3 EXTREMEPOINTSENUM - Enumerating simpler shapes first

Algorithm 9, namely EXTREMEPOINTSENUM, Enumerates closed convex polygon patterns in a bottom-up fashion. Unlike EXTCBO Algorithm, this algorithm allows to prune the search space w.r.t. the **shape complexity** of the description $d \in \mathcal{D}$, that is its number of edges or equivalently the number of linear inequalities in its $\mathcal{H}$-representation. Indeed, the simpler the form of d the better by principle of parsimony.

In order to enumerate all the polygons from simpler shapes to more complex ones, one needs to ensure that if a point $g \in \mathcal{G} \backslash A$ is added to the current extent $A \in \mathbb{P}_{ext}$, the extreme points of the new extent $ext \circ int(A \cup \{g\})$ need to verify the following property:

$$ex(ext \circ int(A \cup \{g\})) = ex(A) \cup \{g\}$$

An object $g \notin A$ verifying the aforementioned property is said to be a **candidate point** for $A \in \mathbb{P}_{ext}$. In what follows, for two distinct objects $g_i, g_j \in \mathcal{G}$, we denote by $\overrightarrow{g_i g_j}^+$, $\overrightarrow{g_i g_j}^-$ and $\overrightarrow{g_i g_j}^0$ elements of $\mathcal{G}$ falling in these regions rather than the regions themselves (see Definition 5.14).

Computing the candidate points. Consider a closed convex polygon pattern $d = (p_i)_{1 \leq i \leq |d|}$. We define a same-size tuple n_d called **candidate points sequence** where $n_d[i]$ gives the set of points that are *visible from and only from* the edge $\overrightarrow{p_i p_{i+1}}$ (see Definition 5.14). Formally, if $|d| \geq 2$:

$$n_d[i] = \overrightarrow{p_i p_{i+1}}^- \cap \left(\bigcap_{j \neq i} \overrightarrow{p_j p_{j+1}}^+ \right) \tag{5.1}$$

Notice that here the + and − on the index of extreme points are done in a cyclic way i.e. if $i = |d|$ then $i+1 = 1$, $i+2 = 2$. If $i = 1$ then $i-1 = |d|$ (i.e. $(a+b = (a+b) \textbf{ mod } |d|)+1$). Note also that if $|d| = 1$ then $n_d[1] = \mathcal{G} \backslash \{p_1\}$. A simpler formula $n_d[i]$ when $|d| \geq 2$ is given below:

$$n_d[i] = \overrightarrow{p_i p_{i+1}}^- \cap \overrightarrow{p_{i-1} p_i}^+ \cap \overrightarrow{p_{i+1} p_{i+2}}^+ \tag{5.2}$$

Extending a pattern and candidate points maintenance. As we have said before, adding a candidate point will add only one extreme points. Interestingly, for $d = (p_i)_{1 \leq i \leq d}$ which extent is

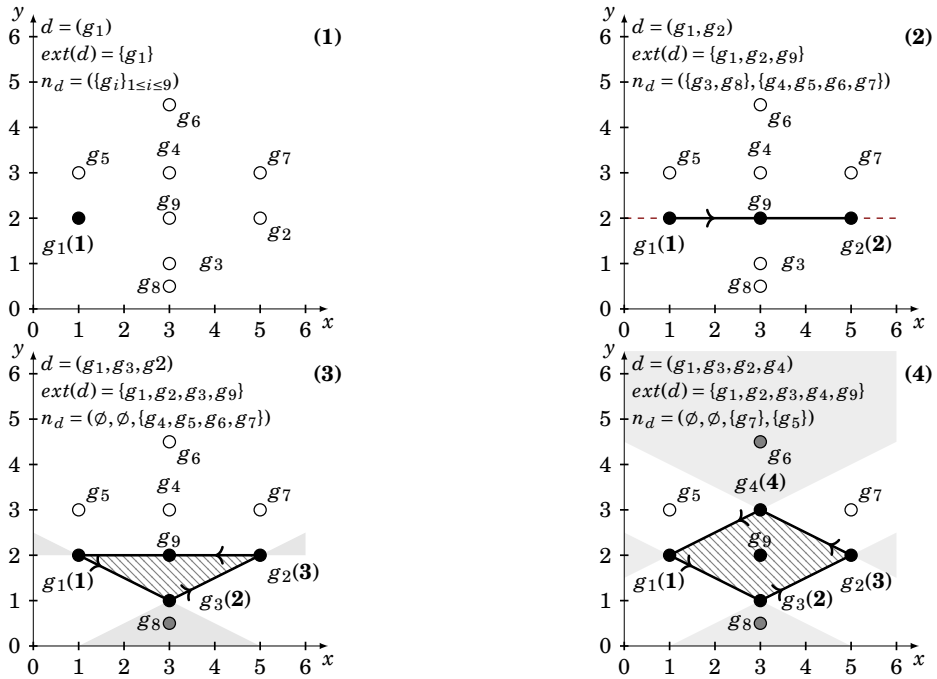


Figure 5.12: EXTREMEPOINTSENUM algorithm step-by-step enumeration.

A the representation of the new polygon after adding candidate $q \in n_d[i]$, i.e. $c = int(A \cup \{q\})$, is given by: $(p_1, ..., p_i, q, p_{i+1} ..., p_{|d|})$. That is no effort is demanded to get the representation of the new created pattern. Still, one need to recompute the newly covered objects, i.e. $ext(c)$. To do so, one can follow the following steps:

$$
\begin{aligned}
ext(c) &= ext(d) \cup ext((p_i, q, p_{i+1})) \\
n_e[i] &= n_d[i] \cap \overrightarrow{p_i q}^- \cap \overrightarrow{q p_{i+1}}^+ \\
n_e[i+1] &= n_d[i] \cap \overrightarrow{q p_{i+1}}^- \cap \overrightarrow{p_i q}^+ \\
n_e[i-1] &= n_d[i-1] \cap \overrightarrow{p_i q}^+ \\
n_e[i+2] &= n_d[i+1] \cap \overrightarrow{q p_{i+1}}^+ \\
n_e[i+k] &= n_d[i+k-1] \textbf{ for } 3 \le k \le |d| - i + 1 \\
n_e[i-k] &= n_d[i-k] \textbf{ for } 2 \le k < i
\end{aligned}
$$

Note that $ext((p_i, q, p_{i+1}))$ is given by the set of objects enclosed in the area formed by the triangle (p_i, q, p_{i+1}). Formally: $ext((p_i, q, p_{i+1})) = \overrightarrow{p_i q}^0 \cup \overrightarrow{q p_{i+1}}^0 \cup \overrightarrow{p_i p_{i+1}}^0 \cup \left(\overrightarrow{p_i p_{i+1}}^- \cap \overrightarrow{p_i q}^+ \cap \overrightarrow{q p_{i+1}}^+\right)$.

Example 5.15. Fig. 5.11 shows a convex polygon $d = (g_1, g_2, g_3, g_4, g_5, g_6)$ with candidate points given by $n_d = (\emptyset, \emptyset, \{g_8\}, \{g_9\}, \emptyset, \emptyset)$. More generally, every point that falls in the *gray* zones is a candidate point for the oriented segment $\overrightarrow{g_3 g_4}$ or $\overrightarrow{g_4 g_5}$. However, the point g_{10} is not a candidate point. Indeed, g_{10} sees two edges $\overrightarrow{g_3 g_4}$ and $\overrightarrow{g_4 g_5}$. □

Algorithm 9: EXTREMEPOINTSENUM

Input: $(\mathcal{G}, \mathcal{M})$ a finite numerical datasets with $(\forall i, j \in \mathcal{G})$ if $i \neq j$ then $\mathcal{M}(i) \neq \mathcal{M}(j)$,
Objects in $\mathcal{G} = \{1, ..., |\mathcal{G}|\}$ totally ordered with $\leq$

Output: Elements of $\mathbb{P}_{ext}$ with $\mathbb{P}$ is the convex set pattern structure (see Definition 5.15)

1 **procedure enumerate** (A,d,n,pos,S)
2 **Print**(A) // Output extent A
3 **for** $k \in 1, ..., |d|$ **do**
4 **for** $j \in n[k]$ ***and*** $j >= pos$ **do**
5 $s_1 \leftarrow S[d[k]][j]$ // first new segment $\overrightarrow{d[k]j}$
6 $s_2 \leftarrow S[j][d[k+1]]$ // second new segment $\overrightarrow{jd[k+1]}$
7 $d_{new} \leftarrow (d[1], ..., d[k], j, d[k+1], ..., d[|d|])$
8 $A_{new} \leftarrow A \cup s_1^0 \cup s_2^0 \cup \left(n[k] \cap s_1^+ \cap s_2^+\right)$
9 $n_{new} \leftarrow (n[1], ..., n[k], n[k], n[k+1], ..., n[|d|])$
10 $n_{new}[k] \leftarrow n[k] \cap s_1^- \cap s_2^+$
11 $n_{new}[k+1] \leftarrow n[k] \cap s_2^- \cap s_1^+$
12 $n_{new}[k-1] \leftarrow n[k-1] \cap s_1^+$
13 $n_{new}[k+2] \leftarrow n[k+1] \cap s_2^+$
14 **enumerate**(A_{new},d_{new}, n_{new},$j+1$,S)
15 **procedure** EXTREMEPOINTSENUM()
16 **Print**($\emptyset$) // Output $\emptyset \in \mathbb{P}_{ext}$
17 Enumerate in *BFS-fashion* all singleton $\{g\} \subseteq \mathcal{G}$
18 Compute **segment index** S
19 **for** $i \in 1, ..., |\mathcal{G}| - 1$ **do**
20 **for** $j \in i+1, .., |\mathcal{G}|$ **do**
21 $s \leftarrow S[i, j]$ // $S[i, j] := \overrightarrow{ij}$
22 **enumerate**($ext(s)$,(i, j), (s^-, s^+),$j+1$,S)

Algorithm EXTREMEPOINTSENUM**.** Algorithm 9, namely The algorithm EXTREMEPOINTSENUM, follows almost the same enumeration principle as that of EXTCBO. The difference lies in maintaining the list of **object candidates** that can be added to a pattern d. The other difference is that the two first levels are enumerated in *Breadth-first search (BFS)* fahsion (Lines 16-17). This allows one to build the **segment index** S that for each object pair $(i, j) \in \mathcal{G} \times \mathcal{G}$ s.t. $i \neq j$ stores the extent $\overrightarrow{ij}^0$, its candidates $\overrightarrow{ij}^-$ and $\overrightarrow{ij}^+$. The algorithm continues in a *DFS-fashion* to enumerate higher levels, i.e. patterns with more than 2 extreme points. To prevent redundancy, when extending a pattern d with a candidate j, only candidate points l s.t. $j < l$ are allowed to be added in the next steps (see pos parameter in line 4).

Example 5.16. Fig. 5.12 gives a detailed step-by-step partial enumeration. In each subfigure, black points belong to the extents, white points are candidate points and gray points are not candidates (points located in the white zones). □

5.5.5 Empirical Evaluation

We report an experimental study of the different algorithms, carried out on a machine equipped with Intel Core i7-2600 CPUs 3.4 Ghz machine with 16 GB RAM. All materials are available on `https://github.com/BelfodilAimene/MiningConvexPolygonPatterns`.

5.5.5.1 Mining polygon patterns

We compare the three algorithms without any constraint: ExtCbO, DelaunayEnum and ExtremePointsEnum. Figure 5.13 plots for each one their run times and the number of pattern candidates they generated. Datasets consist of n objects drawn from the IRIS dataset uniformly from the three different classes for the attributes sepal-length and sepal-width (or petal-lentgh and petal-width). First, notice that ExtCbO generates a lot of candidates discarded by the canonicity test (redundant), while the two others generate each pattern only once. This implies that ExtCbO is from one to two orders of magnitude slower (it is the only one computing closures). Interestingly, ExtremePointsEnum is faster than DelaunayEnum as it does not require to compute and update a Delaunay triangulation (even when the state-of-the-art [156] is used).

5.5.5.2 Impact of the constraints

ExtCbO enumerates convex polygons in a bottom-up fashion w.r.t. inclusion. It can thus only handle maximum perimeter and area constraints that are monotone (the proof is given, e.g. by [26]): when a pattern is generated and does not satisfy the constraint, the algorithm backtracks (a well-known property in pattern mining). DelaunayEnum enumerates convex polygons in a top-down fashion w.r.t. inclusion. It can thus naturally prune w.r.t. minimal support, area and perimeter. ExtremePointsEnum enumerates patterns by inclusion but also from simpler to more complex shapes (extreme points inclusion). It can thus handle maximum shape complexity, perimeter and area constraints.

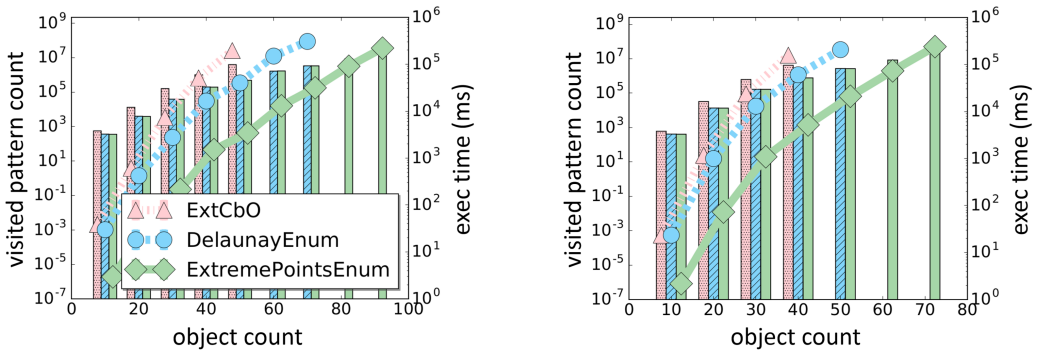
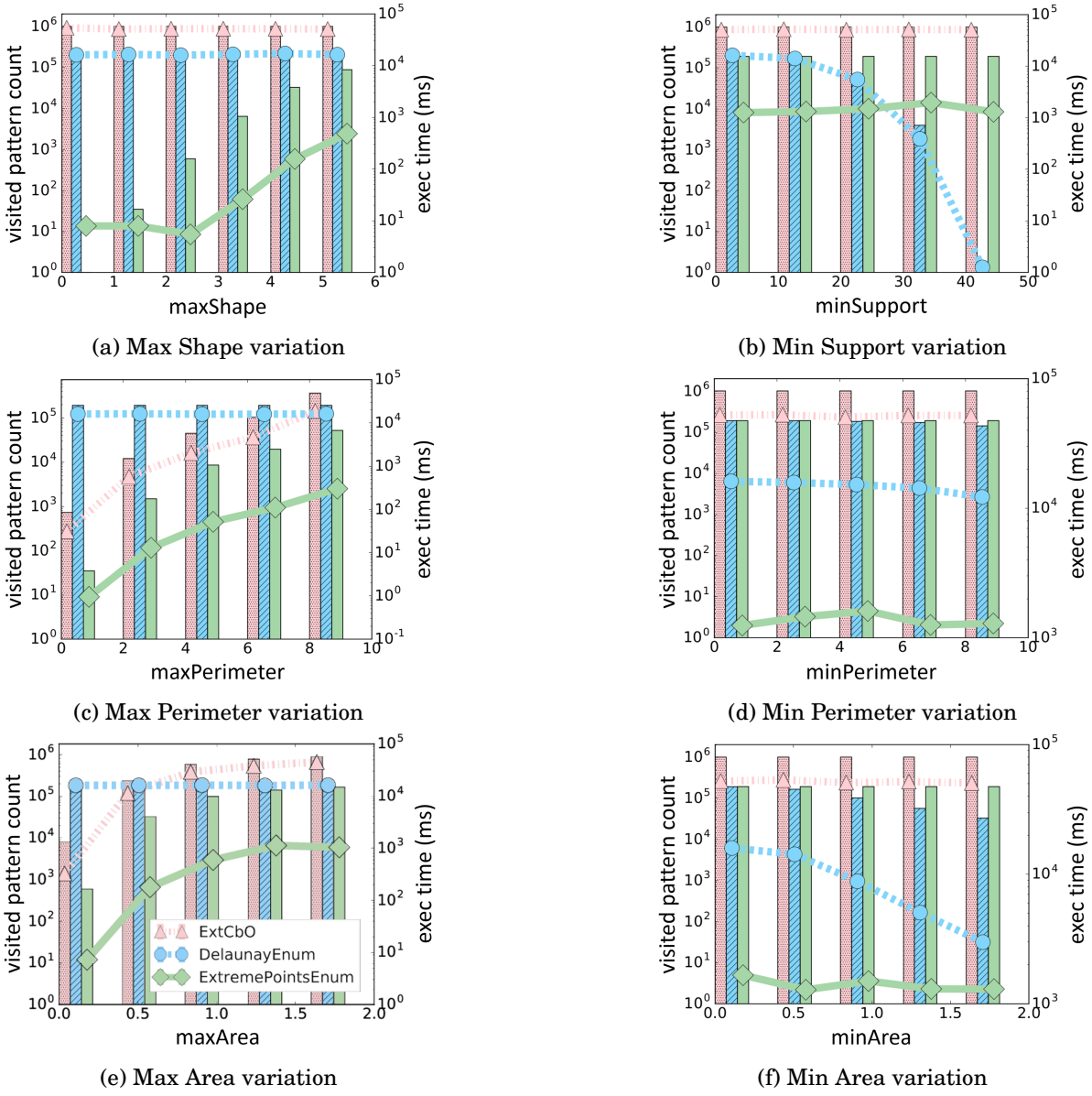


Figure 5.13: Polygon pattern enumeration performance comparison. IRIS sepal-length × sepal-width (left). IRIS petal-length × petal-width (right).



(a) Max Shape variation

(b) Min Support variation

(c) Max Perimeter variation

(d) Min Perimeter variation

(e) Max Area variation

(f) Min Area variation

Figure 5.14: Run time and generated patterns count for our three algorithms when introducing constraints.

Figure 5.14 reports the run time of our three algorithms on the IRIS Sepal length vs. Sepal Width dataset when introducing each constraint separately and varying the associated threshold (min. and max. perimeter are computed as they have the same behavior as min. and max. area, respectively). It also reports the number of generated patterns: The lower, the better. Some algorithms output more patterns as they cannot efficiently handle the constraints (such invalid patterns need to be removed during post-processing). As such, depending on the constraints the user is interested in, one algorithm may be preferred to another.

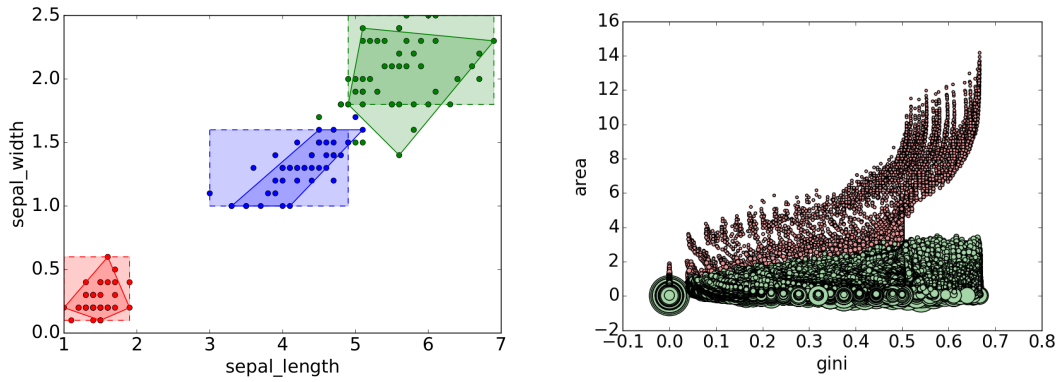


Figure 5.15: **(Left)** Comparing interval and convex polygon top-3 patterns. **(Right)** Comparing interval (red) and convex polygon patterns (green) gini, area and density (represented by points diameter).

5.5.5.3 Intervals vs. convex polygons

Our main motivation for introducing convex polygon patterns is to discover shapes with high density and area, and possibly with high class homogeneity (e.g., low Gini). Figure 5.15 (left) considers the IRIS dataset (Sepal length vs. Sepal Width). It presents the three most frequent polygons that have null Gini, and either 3 or 4 extreme points for a fair comparison: convex polygons better stick to the data without extremely over-fitting.

We also compare interval and polygon patterns in several datasets and plot their area, density and Gini. Figure 5.15 (right) plots all discovered patterns area (Y), Gini (X), and density (point diameter). It appears that convex polygons enable to find shapes with higher density, yet smaller area, over the same Gini range. Rectangles with high area are exactly those that we want to avoid for spatial data: they have high chances to enclose both zones of high and low density, and high impurity (high gini).

5.5.6 Conclusion

In this section we have investigated the problem of enumerating exhaustively and non redundantly separable sets by the language of convex sets, or more particularly convex polygons. We have seen that this language is equivalent to the language of finite conjunction of linear inequalities making the language fare more expressive than the language of intervals discussed in the precedent section. However, the expressivity of this pattern language comes with a cost, its intelligibility. Indeed, in d-dimensional finite numerical datasets $(\mathcal{G}, \mathcal{M})$, while the vertex representation has a maximal size of $|\mathcal{G}|$, the size of the halfspace representation (the number of the linear inequalities) has the order of magnitude of $|\mathcal{G}|^{\lfloor d/2 \rfloor}$ making the intelligibility of such kind of patterns questionable. Moreover, the number of extents could be $2^{|\mathcal{G}|}$ when the points are co-spherical. Other simplifications of the language could be investigated as for instance the language of neighborhood patterns [86] which, sadly, does not induce a pattern structure.

5.6 Conclusion

In this chapter, we have investigated the problem of enumerating definable sets in a finite pattern setup for the particular case of pattern structures and formal contexts. We have extended the state-of-the-art of enumeration techniques by proposing: (1) Algorithm CbOI for concepts enumeration in formal context that leverages existing (inherent) implications between the context attributes [16] and (2) Three algorithms for enumerating definable sets for the particular language of convex sets [20].

However, in this chapter, we have still not discussed Problem 5.1 when non pattern structures are considered. Let us investigate for instance the sequential pattern language [6]. This language does not induce a pattern structure [48] but still induces a pattern multistructure. Hence, according to Theorem 3.5, all definable sets can be obtained from support-closed patterns. Therefore, a **first approach** to enumerate all possible definable sets is to enumerate support-closed patterns [166]. However, since two support-closed patterns can have the same extent (see Section 3.5.1.2), such algorithms are complete and sound but potentially redundant. A **second approach** used in the literature when dealing with pattern setups that are not pattern structures is to use completions in order to transform them to pattern structures (see section 3.7.1). The common used technique is the antichain completion as it is the case for sequential patterns [39, 48] and graph patterns [78]. After such a transformation, algorithms solving Problem 5.3 (e.g Algorithm 6) can be used. However, one needs to keep in mind that the considered language after completion is no longer the same, i.e. it is the logical conjunction of the basic patterns. Hence, using such a technique makes it possible to produce algorithms that are complete and non redundant but not necessarily sound, i.e. some output definable sets are not induced by the basic pattern language.

Whether the first or the second approach is used to tackle the general Problem 5.1, one can correct the different algorithms by:

- Obtaining non-redundancy for the first approach by storing in memory all the already generated extents then checking before an output if the extent has already been output. Such a solution is costly in memory.
- Obtaining soundness for the second approach by checking before an output of a set A if there exists at least one maximal common description $cov^*(A)$ which extent is A. If not so, the algorithm does not output the set A. Such a solution can be costly in time since the number of extents in the completion can be exponential to the number of extent in the basic language (see Example 3.31).

Proposing a better algorithm solving Problem 5.1 remains an open problem that we are thoroughly investigating currently. Our main intuition is to use upper-approximation extents to jump between definable sets. Moreover, when the number of maximal covering descriptions is not infinite and the considered pattern setup is a pattern multistructure, the computation of upper-approximations can be done using only maximal covering descriptions thanks to Theorem 3.4.

Part III

Discriminative Subgroup Discovery

CHAPTER 6

DISCRIMINATIVE SUBGROUP DISCOVERY

There is a large set of definitions of the task of subgroup discovery in the literature [96, 106, 150, 163]. In a very general way, we define here subgroup discovery as follows:

*"The task of finding a **small** subset of **interesting patterns** in a given **dataset**".*

There is four important terms in this definition. The notions of **dataset** and **pattern** has been presented in Chapter 3 and Chapter 5. The two remaining terms are interpreted below:

Intersting. In the sense that someone needs a formal way to say if a pattern is **more interesting** than another, *an order relation*. Often, the evaluation of the interestingness of a pattern is made through the usage of some **quality measure**, that is a mapping that associates to each pattern a value. The higher is this value, the more interesting is the pattern.

Small. In the sense that someone needs a way to select a small set of interesting patterns from the set of all interesting ones.

While the definition of subgroup discovery is quite large, we will consider here a particular task on subgroup discovery which we call **Discriminative Subgroup Discovery**. The following of this chapter is organized as follows:

- **Section 6.1** presents the task of discriminative subgroup discovery.
- **Section 6.2** presents the notion of relevance theory introduced in [81].
- **Section 6.3** presents the common way used in the literature to evaluate subgroup interestingness via a quality measure.
- **Section 6.4** gives an idea about the different approaches existing in the literature tackling the problem of discriminative subgroup discovery.

6.1 Introduction by Examples

From now on, we will make an abstraction on the pattern language that was discussed in Chapter 3 and the provided dataset. We consider then as an input an arbitrary pattern setup $\mathbb{P} = (\mathcal{G}, (\mathcal{D}, \sqsubseteq), \delta)$. In **Discriminative Subgroup Discovery**, the set of objects $\mathcal{G}$ is partitioned into two sets: the set of positive or target instances $\mathcal{G}^+$ and the set of negative instances $\mathcal{G}^-$. Informally, **Discriminative Subgroup Discovery** strives out to find **subgroups** that discriminate/separate positive instances from the negative ones. We call **subgroup** any extent $S \in \mathbb{P}_{ext}$.

Example 6.1. Fig. 6.1 **(left)** depicts a formal context $\mathbb{P}$ (or equivalently a pattern structure) where $\mathcal{G}^+ = \{g_1, g_2, g_3\}$ and $\mathcal{G}^- = \{g_4, g_5, g_6\}$. The set of all possible subgroups is given $\mathbb{P}_{ext}$ depicted on Fig. 6.1 **(right)**. Recall that each extent $A \in \mathbb{P}_{ext}$, has at least one description inducing it. In pattern structure, one could choose the intent $int(A)$. □

$\mathcal{G}$	m_1	m_2	m_3	$label$
$\mathbf{g_1}$	×	×		+
$\mathbf{g_2}$	×	×		+
$\mathbf{g_3}$	×		×	+
g_4	×	×		−
g_5	×	×		−
g_6		×	×	−

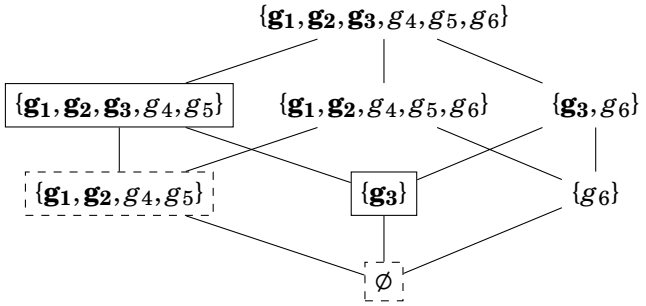


Figure 6.1: Formal Context $\mathbb{P} = (\mathcal{G}, \mathcal{M}, I)$ **(left)** and its subgroups $\mathbb{P}_{ext}$ **(right)**. Objects in **bold** are positive instances.

Example 6.2. Consider now the numerical dataset depicted in Fig. 6.2. The set of subgroups depends on the hypothesis space, i.e. the description language. If we consider the interval pattern structure, the set $\{\mathbf{g_1}, \mathbf{g_2}, \mathbf{g_3}\}$ is a subgroup whose intent is $[1,2] \times [1,3]$. If the language of convex set pattern language is considered, then $\mathcal{G}^+ = \{\mathbf{g_1}, \mathbf{g_2}, \mathbf{g_3}, \mathbf{g_4}\}$ is an extent. □

	m_1	m_2	label
$\mathbf{g_1}$	1	2	+
$\mathbf{g_2}$	1	3	+
$\mathbf{g_3}$	2	1	+
$\mathbf{g_4}$	3	5	+
g_5	3	4	−
g_6	2	5	−

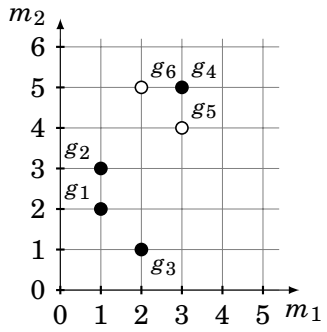


Figure 6.2: A labeled numerical dataset $(\mathcal{G}, \mathcal{M})$ **(left)** and its representation in plane **(right)** where Black (resp. white) dots represent positive (resp. negative) instances.

6.2 Relevance Theory

Relevance theory has been introduced by [81, 82] to compare between subgroups based on the positive and negative instances they cover in the aim of reducing the number of the considered subgroups. Before defining formally what relevance is. Let us start by a toy example.

Example 6.3. Consider the formal context depicted in Fig. 6.1, Let $S = \{\mathbf{g_1}, \mathbf{g_2}, g_4, g_5\}$ and $T = \{\mathbf{g_1}, \mathbf{g_2}, g_4, g_5, g_6\}$. If someone aims to find subgroups that discriminate positive instances in $\mathcal{G}^+$ from negative ones $\mathcal{G}^-$, he should prefer S than T since all positive instances isolated by S are also isolated by T. However, S covers less negative instances than T. We will say that S is **more relevant than** T and introduce formally this notion in Definition 6.1. □

6.2.1 Basic Definitions

Definition 6.1. A subgroup $S \in \mathbb{P}_{ext}$ is said to be **more relevant than** $T \in \mathbb{P}_{ext}$ iff:

$$S \cap \mathcal{G}^+ \supseteq T \cap \mathcal{G}^+ \quad \textbf{and} \quad S \cap \mathcal{G}^- \subseteq T \cap \mathcal{G}^-$$

In other words S encloses all positive instances in T while it does enclose less negative instances.

One could see that **"more relevant than"** induces a partial order on $\mathbb{P}_{ext}$. The maximal elements in $\mathbb{P}_{ext}$ w.r.t. this order are said to be **relevant**.

Definition 6.2. A subgroup $S \in \mathbb{P}_{ext}$ is said to be **relevant** iff: if there exists a subgroup $T \in \mathbb{P}_{ext}$ that is more relevant than $S \in \mathbb{P}_{ext}$ then $T = S$.

Example 6.4. Consider the formal context considered in Fig. 6.1. One can check that the relevant subgroups are those surrounded by a continuous rectangle on the right hand side of the figure, i.e. $\{\mathbf{g_1}, \mathbf{g_2}, \mathbf{g_3}, g_4, g_5\}$ and $\{\mathbf{g_3}\}$.

Consider now the numerical dataset considered in Fig. 6.2, if the interval pattern language is considered then relevant subgroups are $\{\mathbf{g_1}, \mathbf{g_2}, \mathbf{g_3}\}$ and $\mathcal{G}$. If the convex polygon pattern language is considered, only $\mathcal{G}^+$ is relevant. □

Note 6.1. Without loss of generality, we have targeted here positive instances. One can define the same notions of relevance toward negative instances analogously. □

6.2.2 Relevance in Pattern Structures

We have defined relevant subgroups in the general case of pattern setup. We will now investigate some properties that relevant subgroups have when the considered pattern setup $\mathbb{P}$ is a **pattern structure**. From now on, the pattern setup $\mathbb{P}$ is a pattern structure. This section can be seen as a small generalization of works of [82] and [88] for the general case of pattern structures rather than only the language of itemsets.

Let us start by the first important observation formulated in the following proposition.

Proposition 6.1. Subgroup $ext \circ int\left(\mathcal{G}^+\right)$ is relevant.

Proof. Suppose now that $A \in \mathbb{P}_{ext}$ is more relevant than $ext \circ int\left(\mathcal{G}^+\right)$. Then we have:

$$ext \circ int\left(\mathcal{G}^+\right) \cap \mathcal{G}^+ \subseteq A \cap \mathcal{G}^+ \quad \text{and} \quad ext \circ int\left(\mathcal{G}^+\right) \cap \mathcal{G}^- \supseteq A \cap \mathcal{G}^-$$

Since $ext \circ int$ is extensive, we obtain that $A \cap \mathcal{G}^+ = ext \circ int\left(\mathcal{G}^+\right) \cap \mathcal{G}^+ = \mathcal{G}^+$. Hence $\mathcal{G}^+ \subseteq A$. On the other hand, since $A \in \mathbb{P}_{ext}$ then it is a fixpoint of $ext \circ int$. Since $ext \circ int$ is order-preserving then $ext \circ int(\mathcal{G}^+) \subseteq A$. Hence, $ext \circ int\left(\mathcal{G}^+\right) \cap \mathcal{G}^- \subseteq A \cap \mathcal{G}^-$. Therefore, $ext \circ int\left(\mathcal{G}^+\right) \cap \mathcal{G}^- = A \cap \mathcal{G}^-$. We conclude that $A = ext \circ int\left(\mathcal{G}^+\right)$, i.e. $ext \circ int\left(\mathcal{G}^+\right)$ is relevant.

Another important property is the fact that a subgroup is relevant then it is also **closed-on-the-positives**. Let us define first this notion.

Definition 6.3. A subgroup $S \in \mathbb{P}_{ext}$ is said to be **closed-on-the-positives** or ***cotp*** for short iff it is the smallest extent covering its positive instances. Formally, $S = ext \circ int\left(S \cap \mathcal{G}^+\right)$. The set of *cotp* is given then by $ext \circ int\left[\wp\left(\mathcal{G}^+\right)\right]$.

Note 6.2. Please note that the mapping $S \mapsto ext \circ int(S \cap \mathcal{G}^+)$ is technically not a closure operator on $(\wp(\mathcal{G}), \subseteq)$. However, this mapping is still idempotent and order-preserving but not extensive. However and interestingly, this mapping is a kernel operator on $(\mathbb{P}_{ext}, \subseteq)$. The term *closed-on-the-positive* is used for elements in $\wp(\mathcal{G})$ s.t. $S = ext \circ int(S \cap \mathcal{G}^+)$ since: (1) they are closed w.r.t. $ext \circ int$, i.e. they are extents and (2) they have a basis on $\mathcal{G}^+$, i.e. they are generated only by their positive instances. □

Proposition 6.2. If a subgroup $S \in \mathbb{P}_{ext}$ is relevant then it is *cotp*.

Proof. Suppose that S is relevant but not *cotp*, then $S \neq ext \circ int\left(S \cap \mathcal{G}^+\right)$. We have $S \cap \mathcal{G}^+ \subseteq ext \circ int\left(S \cap \mathcal{G}^+\right)$ since $ext \circ int$ is extensive. Moreover, since $S \in \mathbb{P}_{ext}$ is a fixpoint of $ext \circ int$, $S \cap \mathcal{G}^+ \subseteq S$ and $ext \circ int$ is order-preserving, we obtain $ext \circ int\left(S \cap \mathcal{G}^+\right) \subsetneq S$. Hence, $ext \circ int\left(S \cap \mathcal{G}^+\right) \cap \mathcal{G}^+ = S \cap \mathcal{G}^+$ but at the same time $ext \circ int\left(S \cap \mathcal{G}^+\right) \cap \mathcal{G}^- \subsetneq S \cap \mathcal{G}^-$. Therefore, $ext \circ int\left(S \cap \mathcal{G}^+\right)$ is *strictly* more relevant than S which contradicts the fact that S is relevant.

Proposition 6.2 shows that all relevant subgroups are necessarily *cotp*. The converse is however not true as shown in the example below.

Example 6.5. One can verify that the subgroups that are *cotp* are those surrounded by continuous or dashed rectangle. One can check that subgroups that are surrounded by dashed rectangles are *cotp* but not relevant. For instance subgroup $S_1 = \{\mathbf{g_1}, \mathbf{g_2}, g_4, g_5\}$ is *cotp*, since $int(\{\mathbf{g_1}, \mathbf{g_2}\}) = \{m_1, m_2\}$ which extent is S_1. However, S_1 is still strictly less relevant than $R_1 = \{\mathbf{g_1}, \mathbf{g_2}, \mathbf{g_3}, g_4, g_5\}$ which covers the same negative instances of S_1 but encloses an additional positive instance g_3. □

$\mathcal{G}$	m_1	...	m_n	$label$
$\mathbf{g_1}$				
$\vdots$		$\neq$		$+$
$\mathbf{g_n}$				
g_{n+1}				
$\vdots$		$\neq$		$-$
g_{2n}				

Table 6.1: A formal context $(\mathcal{G}, \mathcal{M}, I)$ which is the subposition of two contranominal scales of size $n \in \mathbb{N}$ [80], i.e. $n \geq 2$, $\mathcal{G} = \{g_i\}_{1 \leq i \leq 2 \cdot n}$, $\mathcal{G}^+ = \{g_i\}_{1 \leq i \leq n}$, $\mathcal{G}^- = \{g_i\}_{n+1 \leq i \leq 2 \cdot n}$, $\mathcal{M} = \{m_i\}_{1 \leq i \leq n}$ and $(\forall i \in \{1...2 \cdot n\}, \forall j \in \{1, ..., n\})\ g_i\ I\ m_j \Leftrightarrow j \notin \{i, i-n\}$.

Following the *test of relevance* in [88]. The theorem below gives a characterization of relevant patterns in pattern structures.

Theorem 6.1. A subgroup $S \in \mathbb{P}_{ext}$ is relevant **iff** S is *cotp* and the following property holds:

$$(\forall g^+ \in \mathcal{G}^+ \backslash S) \quad S \cap \mathcal{G}^- \neq ext \circ int\left(S \cup \{g^+\}\right) \cap \mathcal{G}^-$$

Proof. Let us show both implications independently.

Suppose that S is relevant, then according to Proposition 6.2, S is *cotp*. Suppose now that the given property does not hold. That is $\exists g^+ \in \mathcal{G}^+ \backslash S$ s.t. $S \cap \mathcal{G}^- = ext \circ int\left(S \cup \{g^+\}\right) \cap \mathcal{G}^-$. Moreover, it is clear that $S \cap \mathcal{G}^+ \subsetneq \left(S \cup \{g^+\}\right) \cap \mathcal{G}^+ \subseteq ext \circ int\left(S \cup \{g^+\}\right) \cap \mathcal{G}^+$. Hence, $ext \circ int\left(S \cup \{g^+\}\right)$ is strictly more relevant than S which contradicts the fact that S is relevant.

Suppose now that both properties hold but S is not relevant. Hence, there exists at least one $T \in \mathbb{P}_{ext}$ that is *cotp* and strictly more relevant than S. Using the fact that S and T are *cotp* and $S \cap \mathcal{G}^+ \subseteq T \cap \mathcal{G}^+$ we obtain $S = ext \circ int\left(S \cap \mathcal{G}^+\right) \subseteq ext \circ int\left(T \cap \mathcal{G}^+\right) = T$.

If $S \cap \mathcal{G}^+ = T \cap \mathcal{G}^+$ then $S = T$ which is contradictory since T is strictly more relevant than S. Hence, $S \cap \mathcal{G}^+ \subsetneq T \cap \mathcal{G}^+$. Moreover, we have $T \cap \mathcal{G}^- = S \cap \mathcal{G}^-$ since $S \subseteq T$ and $T \cap \mathcal{G}^- \subseteq S \cap \mathcal{G}^-$. Let $g^+ \in (T \backslash S) \cap \mathcal{G}^+$ (note that $(T \backslash S) \cap \mathcal{G}^+$ is not empty). We have $S \subseteq S \cup \{g^+\} \subseteq T$. Hence, since T, S are fixpoints of $ext \circ int$ we obtain: $S \subseteq ext \circ int\left(S \cup \{g^+\}\right) \subseteq T$. Thus, $ext \circ int\left(S \cup \{g^+\}\right) \cap \mathcal{G}^- = S \cap \mathcal{G}^-$. This contradicts the fact that S has the second property. ∎

6.2.3 Discussion

We have seen that the notion of relevance (to the positive label) allows to reduce the number of considered subgroups when the task is to find subgroups separating positive instances from the negative ones. Interestingly, when the positive instances are *separable* by the pattern language, i.e. $\mathcal{G}^+ \in \mathbb{P}_{ext}$, there is a unique relevant subgroup given by $\mathcal{G}^+$. Still, as observed in Example 6.6,

someone need to keep in mind that, in some cases, all closed-on-the-positives are relevant making the number of relevant subgroups possibly exponential to the size of $\mathcal{G}$.

Example 6.6. Table 6.1 depicts a formal context (or equivalently a pattern structure) where one could check that there is 2^n *cotp* subgroups and all these *cotp* subgroups are relevant. □

6.3 Discriminative Subgroups Evaluation

We have seen a way to compare between subgroups through relevance theory. However, the number of relevant subgroups could be still exponential to the number of objects as stated in Example 6.6. Quality measure is another tool to compare between subgroups regarding the number of positive and negative instances covered by them.

Since discriminative subgroups to the positive class can be seen as *class association rules (CARs)* [121] where the consequent is fixed to the positive class, the quality measures evaluating them comes directly from association rules evaluation literature. A plethora of quality measures has been proposed and studied in the literature for such a purpose [14, 52, 73, 84, 95, 101, 120, 154].

6.3.1 Basic Definitions

From now on, $\mathbb{P} = (\mathcal{G},(\mathcal{D},\sqsubseteq),\delta)$ denotes a finite pattern setup, i.e. $\mathcal{G}$ is finite and the set of objects $\mathcal{G}$ is partitioned into $\{\mathcal{G}^+,\mathcal{G}^-\}$. We call **positive prevalence**, the quantity denoted α and given by the proportion of positive instances $\alpha = |\mathcal{G}^+|/|\mathcal{G}|$. The **negative prevalence** is defined dually and is given by $1-\alpha$. Let us define now what a quality measure is.

Definition 6.4. We call a **quality measure** any mapping $\varphi : \mathcal{D} \to \mathbb{R}$ that associate to each pattern in $\mathcal{D}$ a value in $\mathbb{R}$. A quality measure $\varphi : \mathcal{D} \to \mathbb{R}$ is said to be **extent-based** iff:

$$(\forall c,d \in \mathcal{D})\ ext(c) = ext(d) \Longrightarrow \varphi(c) = \varphi(d)$$

We will be interested here by *extent-based quality measures*. The most basic extent-based measure is the **support**, i.e. $d \mapsto |ext(d)|$ that associate to each pattern the size of its extent. Generally, extent-based measures can be defined by the help the **relative support** defined below.

Definition 6.5. The **relative support** measure is given by:

$$relsup : \wp(\mathcal{G}) \times \wp(\mathcal{G}) \to \mathbb{R}, (S,R) \mapsto \frac{|S \cap R|}{|R|}$$

Given a set of objects $S \subseteq \mathcal{G}$ and reference set of objects $R \subseteq \mathcal{G}$, $relsup(S,R)$ evaluates the presence of S in R.

If we are interested only by judging if a subgroup is better than another by the mean of quality measure, then two distinct quality measures ordering in the same way the set of all

subgroups should be considered indistinguishably. We talk here about the notion of measure compatibility [73] defined below.

Definition 6.6. Two quality measures ϕ_1, ϕ_2 are said to be **compatible** iff:

$$(\forall c, d \in \mathcal{D})\ \phi_1(c) < \phi_1(d) \Longleftrightarrow \phi_2(c) < \phi_2(d)$$

Note 6.3. As stated in [73], Compatible quality measures ϕ_1, ϕ_2 are **equality-preserving**, i.e. $(\forall c, d \in \mathcal{D})\ \phi_1(c) = \phi_1(d) \Longleftrightarrow \phi_2(c) = \phi_2(d)$. □

There are many ways to express the formula of quality measure ϕ. Following [73, 84], common quality measures depend on the contingency table, the number of covered positive and negative instance in and outside the subgroup. Such measures are said to be **probability-based quality measures** [84]. They can be expressed hence as function on the *Receiver Operating Characteristic (ROC) space* (see for example [73, 101]). In other words, the quality measures can be expressed using the true positive rate and the false positive rate defined below.

Definition 6.7. The **true positive rate** (resp. **false positive rate**) measure, is the quality measure denoted tpr (resp. fpr) and given by:

$$tpr : d \mapsto relsup\left(ext(d), \mathcal{G}^+\right) = \frac{\left|ext(d) \cap \mathcal{G}^+\right|}{\left|\mathcal{G}^+\right|} \qquad fpr : d \mapsto relsup\left(ext(d), \mathcal{G}^-\right) = \frac{\left|ext(d) \cap \mathcal{G}^-\right|}{\left|\mathcal{G}^-\right|}$$

Now that we have defined both true positive rate and false positive rate. Let us define formally what is probability-based quality measures.

Definition 6.8. A quality measure ϕ is said to be **probability-based** iff:

$$(\forall c, d \in \mathcal{D})\ tpr(c) = tpr(d) \text{ and } fpr(c) = fpr(d) \text{ then } \phi(c) = \phi(d)$$

That is ϕ can be seen as a function of tpr and fpr when $\mathcal{G}$ and its partition $\{\mathcal{G}^+, \mathcal{G}^-\}$ is fixed.

Note 6.4. From now on and for the sake of brevity, when we say ϕ is a quality measure we mean *probability-based quality measures*. Moreover, it is clear that any probability-based quality measure is also extent-based. Therefore, depending on the context, we will use interchangeably (1) $\phi : \mathcal{D} \to \mathbb{R}$ as a mapping which domain is the description space, (2) $\phi : \wp(\mathcal{G}) \to \mathbb{R}$ as a mapping which domain is set of objects, i.e. evaluates subgroups and (3) $\phi : [0,1] \times [0,1] \to \mathbb{R}$ as a mapping in the *ROC* space where the x-axis (resp. y-axis) represents the false (resp. true) positive rate. In other words, $\phi(d) = \phi(ext(d)) = \phi(fpr(d), tpr(d))$. □

We give in Table 6.2 different (*probability-based*) quality measures expression. Note that each block of measure represents compatible measures following Definition 6.6. For instance there is 13 measures stated in Table 6.2 that are compatible with the precision/confidence (including the growth rate) as observed in [97]. We draw also the reader attention to the parametric $Kl\ddot{o}sgen_\omega$ measure where the parameter $\omega \in [0, +\infty)$. This measure is equal to the *Weighted relative accuracy (WRAcc)* [114] if $\omega = 1$, equal to the added value if $\omega = 0$ and equal to the *binomial test* if $\omega = 0.5$ [126]. Notice that sometimes measures are not defined when the true positive rate or the false positive rate are equal to the extreme values 0 or 1.

6.3.2 Measure Properties

The literature abounds with axiomatization of the properties that a measure can have [84, 120, 141]. We will not explore in this sections in details all the measure properties, we will state two important ones that we will use in Chapter 7.

Definition 6.9. A probability-based quality measure $\phi : [0,1] \times [0,1] \to \mathbb{R}$ is said to have:

(W) The Weak Dominance Property. iff $\forall x_1, x_2, y_1, y_2 \in [0,1]$, we have:

$$x_1 \geq x_2 \text{ and } y_1 \leq y_2 \implies \phi(x_1, y_1) \leq \phi(x_2, y_2)$$

(S) The Strong Dominance Property. iff $\forall x_1, x_2, y_1, y_2 \in [0,1]$, we have:

$$(x_1 \geq x_2 \text{ and } y_1 < y_2) \textbf{ or}$$
$$(x_1 > x_2 \text{ and } y_1 \leq y_2) \implies \phi(x_1, y_1) < \phi(x_2, y_2)$$

Please recall that x (resp. y) represents the false (resp. true) positive rate, i.e. $(\forall d \in \mathcal{D})\ \phi(d) := \phi(fpr(d), tpr(d))$. The word *dominance* is used here to refer to the fact that if a point (x_2, y_2) dominates another point (x_1, y_1) in the ROC-space then the quality of (x_2, y_2) with a measure having a dominance property is better than the quality of (x_1, y_1).

Note 6.5. Clearly, a probability-based quality measure having the strong dominance property has also the weak one. Moreover, compatible-measures have the same dominance properties. □

A discriminative pattern d can be seen as a class association rule where the premise is d and the conclusion is fixed to the positive class, i.e. $d \Rightarrow +$. Piatetsky-Shapiro axiomatized association rules properties in [141]. We recall below properties **(P2)** and **(P3)** for our particular case.

Definition 6.10. Let $\alpha := |\mathcal{G}^+|/|\mathcal{G}|$ be the (fixed) positive prevalence, a probability-based quality measure $\phi : [0,1] \times [0,1] \to \mathbb{R}$ is said to have:

(P2) The **Piatetsky-Shapiro (P2) property** iff ϕ increases whenever the true positive rate increases and the support of the pattern remains the same. Formally, $\forall x_1, x_2, y_1, y_2 \in [0,1]$, we have:

$$\alpha \cdot y_1 + (1-\alpha) \cdot x_1 = \alpha \cdot y_2 + (1-\alpha) \cdot x_2 \textbf{ and } y_1 \leq y_2 \implies \phi(x_1, y_1) \leq \phi(x_2, y_2)$$

(P3) the **Piatetsky-Shapiro (P3) property** iff ϕ decreases whenever the support of the pattern increases and the true positive rate remains the same. Formally, $\forall x_1, x_2, y_1, y_2 \in [0,1]$, we have:

$$\alpha \cdot y_1 + (1-\alpha) \cdot x_1 \geq \alpha \cdot y_2 + (1-\alpha) \cdot x_2 \textbf{ and } y_1 = y_2 \implies \phi(x_1, y_1) \leq \phi(x_2, y_2)$$

Note 6.6. Please notice that $\alpha \cdot y + (1-\alpha) \cdot x$ represents the support of a description whose true positive rate is y and false positive rate is x. □

Proposition 6.3. If a probability-based quality measure has the weak dominance property then it has both the the Piatetsky-Shapiro **(P2)** and **(P3)** properties.

Proof. Let be a quality measure ϕ having the weak dominance property. Let us show that ϕ has both Piatetsky-Shapiro **(P2)** and **(P3)** properties. Let $x_1, x_2, y_1, y_2 \in [0,1]$:

- Suppose that $\alpha \cdot y_1 + (1-\alpha) \cdot x_1 = \alpha \cdot y_2 + (1-\alpha) \cdot x_2$ **and** $y_1 \leq y_2$. It is easy to see that under this hypothesis, we have $x_1 \geq x_2$. Using the the weak dominance property, we conclude $\phi(x_1, y_1) \leq \phi(x_2, y_2)$. Thus, ϕ has the Piatetsky-Shapiro **(P2)** property.
- Suppose that $\alpha \cdot y_1 + (1-\alpha) \cdot x_1 \geq \alpha \cdot y_2 + (1-\alpha) \cdot x_2$ **and** $y_1 = y_2$. It is easy to see that under this hypothesis, we have $x_1 \geq x_2$. Using the the weak dominance property, we conclude $\phi(x_1, y_1) \leq \phi(x_2, y_2)$. Thus, ϕ has the Piatetsky-Shapiro **(P3)** property.

This concludes the proof. ∎

Note 6.7. Please note that the converse of Proposition 6.3 is not true. Indeed, consider the following measure: $\phi : x, y \mapsto 1 - \frac{\alpha}{2} \cdot y - (1-\alpha) \cdot x$. One could easily show that ϕ has both Piatetsky-Shapiro **(P2)** and **(P3)** properties by noticing that $\phi : x, y \mapsto 1 - \left(\alpha \cdot y + (1-\alpha) \cdot x\right) + \frac{\alpha}{2} \cdot y$. Yet, ϕ has not the the weak dominance property.

We give in Table 6.2 the dominance property for each quality measure. We have:

- The parametric $Kl\ddot{o}sgen_\omega$ measure has even not the weak dominance property when $\omega > 1$.
- All measures having the *weak dominance property* in Table 6.2, have, *a priori*, the *strong one* when the true positive rate range or the false positive rate range is modified. For instance, the F_β score has the strong dominance property when the true positive rate is strictly positive. Discriminativity measure has the strong dominance property when the true positive rate is strictly positive and the false positive rate is strictly below 1.
- There is two common measures in the literature, **Gini Index** and the **Chi-Squared** measures that have not the weak dominance property. These measures do not target only subgroups with a high coverage on positive instance and a small coverage on the negative instances, but evaluate the purity of the subgroup or equivalently the deviation from the independent case, i.e. $tpr = fpr$.

Last but not least, Proposition 6.4 presents a tight link between relevance theory and the dominance property.

Proposition 6.4. Let $c, d \in \mathcal{D}$ be two patterns and let ϕ be a probability-based quality measure having the weak dominance property. If $ext(c)$ is more relevant than $ext(d)$ then $\phi(c) \geq \phi(d)$

Proof. The proof is trivial using Definition 6.1 and Definition 6.9. ∎

Measure	**Definition**	**Domi.**
False Positive Rate	x	
True Positive Rate	y	w
Prevalence	α	w
Support/Coverage [4]	$s := \alpha \cdot y + (1-\alpha) \cdot x$	
Precision/Confidence [4]	$p := \alpha \cdot y/s$	w
Growth Rate/Odd Multiplier [57]	$[(1-\alpha)/\alpha] \cdot [p/(1-p)] = y/x$	w
Ganascia index [97]	$2p-1$	w
Sebag-Schoenaur [97]	$p/(1-p)$	w
Example/counter-example rate [97]	$2-1/p$	w
Ohsaki's Conviction [135]	$(1-\alpha)^2/(1-p)$	w
Brin's Interest/Lift/Strength [97]	p/α	w
Mutual information [97]	$log(p/\alpha)$	w
One way support [84]	$p \cdot log(p/\alpha)$	w
Added Value [114]	$p-\alpha$	w
Certainty Factor/Loevinger [97]	$(p-\alpha)/(1-\alpha)$	w
Brin's Conviction [97]	$(1-\alpha)/(1-p)$	w
Zhang [171]	$\left(1-\frac{\alpha}{p}\right)/\left(max\left(\frac{\alpha}{p},1\right)-\alpha\right) = (y-x)/max(y,x)$	w
Guillaume-Khenchaff [65]	$(p-\alpha)/\alpha$ **IF** $p < \alpha$ **ELSE** $(p-\alpha)/(1-\alpha)$	w
Odds Ratio [154]	$\Omega := [y \cdot (1-x)]/[x(1-y)]$	w
Yule's Q [154]	$(\Omega-1)/(\Omega+1)$	w
Yule's x [154]	$\left(\sqrt{\Omega}-1\right)/\left(\sqrt{\Omega}+1\right)$	w
Least contradiction [84]	$[\alpha \cdot y - (1-\alpha) \cdot x]/\alpha$	**S**
Accuracy [84]	$\alpha \cdot y - (1-\alpha) \cdot x + (1-\alpha)$	**S**
Weighted Relative Accuracy [114]	$\alpha \cdot (1-\alpha) \cdot (y-x)$	**S**
Informedness [15]	$y-x$	**S**
Binomial Test [125]	$\alpha \cdot (1-\alpha) \cdot (y-x)/\sqrt{s}$	**S**
Klösgen$_\omega$ ($\omega \in [0,1]$) [101]	$\alpha \cdot (1-\alpha) \cdot s^{\omega-1} \cdot (y-x)$	**S**
Klösgen$_\omega$ ($\omega \in (1,+\infty)$) [101]	$\alpha \cdot (1-\alpha) \cdot s^{\omega-1} \cdot (y-x)$	
Linear correlation coefficient [154]	$\phi := \sqrt{[(\alpha \cdot (1-\alpha))/(s \cdot (1-s))]} \cdot (y-x)$	**S**
Cohen's kappa (κ) [154]	$\kappa := 2\alpha \cdot (1-\alpha)(y-x)/[\alpha + (1-2\alpha) \cdot s]$	**S**
Cosine/G-Measure [84]	$y/\sqrt{[y + (\{1/\alpha\}-1) \cdot x]}$	**S**
m-estimate ($m \in (0,+\infty)$) [73]	$\alpha \cdot [y + m/n]/[\alpha \cdot y + (1-\alpha) \cdot x + m/n]$	**S**
Discriminativity [30]	$n^2 \cdot \alpha \cdot (1-\alpha) \cdot y \cdot (1-x)$	w
F_β score ($\beta \in [0,+\infty)$) [101]	$[(1+\beta^2) \cdot y]/[y + (\{1/\alpha\}-1) \cdot x + \beta^2]$	w
Gini Index [1]	$2 \cdot \alpha \cdot (1-\alpha) \cdot \phi^2$	
Chi-Squared (χ^2) [1]	$\chi^2 := n \cdot \phi^2$	

Table 6.2: Quality of a subgroup S using its *false positive rate* x, its *true positive rate* y, positive prevalence α, size of the set of objects $|\mathcal{G}| = n$. The **Domi.** column shows if the measure have the weak dominance property (w), the strong one (**S**) or no property (nothing).

6.4 Discriminative Subgroups Enumeration

Several algorithms have been proposed in the literature to explore the pattern search space in order to provide a concise set of interesting subgroups, for example Top-k subgroups w.r.t. a quality measure ϕ. There are two main categories of subgroup discovery algorithms:

1. **Exact Algorithms** which rely in general on a complete enumeration of the subgroups [10, 89, 103, 118, 119, 125, 163] to output exact solutions of the targeted subgroup discovery task. Several techniques are used to enhance the performance of these exhaustive algorithms as for instance: (1) enumerate only closed (or closed-on-the-positives) patterns rather than all possible patterns to avoid visiting redundantly the same subgroups, i.e the same extents and (2) leverage the properties of the quality measure to prune uninteresting parts of the search space [89, 90, 118].
2. **Non-Exact Algorithms** which propose approximate solutions for the subgroup enumeration task. The literature abounds with such algorithms: Beam Search Techniques [59, 115, 159, 160], Evolutionary Algorithms [44], sampling techniques [23, 30, 31] and anytime algorithms [34].

Note 6.8. One sad fact is that the (exhaustive) enumeration of relevant subgroups has received less interest while at the same time the set of interesting discriminative subgroups is always a subset of the relevant ones. For instance, the best known existing algorithms when dealing with formal contexts were proposed by Grosskreutz in [88]: (1) a POLY-OUTPUT but not PSPACE Algorithm and (2) a PSPACE but not POLY-OUTPUT one. Guyet et al. [92] investigated also lately the problem of relevant subgroups enumeration when the interval patterns are considered. The existence of a POLY-OUTPUT and PSPACE algorithm, or better a POLY-DELAY one, for relevant subgroup enumeration remains an open problem. □

6.4.1 Finding the Top-quality Subgroup

We have seen in Chapter 5 several algorithms that enumerate exhaustively and non-redundantly the set of all extents of a formal context or a pattern structure. We have seen also that, depending on the order of traversal of the search space, e.g. top-down or bottom-up, some constraints could be incorporated easily in the algorithms to avoid visiting some subgroups that does not verify the constraints. We will show here for instance, how a top-down algorithm for enumerating extents in a pattern setup can be easily adapted to a **branch-and-bound algorithm** to look for the best quality subgroup (or the top-k subgroups) thanks to **optimistic estimates** [90, 161]. For instance Algorithm 4 (CBO-TD), Algorithm 5 (CBOI) for formal contexts, Algorithm 7 (MININTCHANGE) for interval pattern structure and Algorithm 8 (DELAUNAYENUM) for convex polygon pattern structure can be adapted to such a branch-and-bound scheme presented in Algorithm 10. Suppose now that we have some finite pattern setup $\mathbb{P} = (\mathcal{G}, (\mathcal{D}, \sqsubseteq), \delta)$ where $\mathcal{G}$ is partitioned into $\{\mathcal{G}^+, \mathcal{G}^-\}$.

6.4.1.1 Optimistic estimates

Definition 6.11. Let ϕ be an extent-based quality measure (see Definition 6.4), we call **optimistic estimates of** ϕ any extent-based quality measure $\phi^{oe} : \wp(\mathcal{G}) \to \mathbb{R}$ s.t.

$$(\forall S, T \in \wp(\mathcal{G}))\ T \subseteq S \implies \phi(T) \leq \phi^{oe}(S)$$

An optimistic estimates is said to be **tight** iff: $(\forall S \in \wp(\mathcal{G}), \exists T \subseteq S)\ \phi^{oe}(S) = \phi(T)$.

Note 6.9. Please note that is a quality measure ϕ is order-preserving, i.e:

$$(\forall S, T \in \wp(\mathcal{G}))\ T \subseteq S \implies \phi(T) \leq \phi(S)$$

then ϕ is its own tight optimistic estimates. □

Interestingly, the following proposition shows that there is an easy way for build a tight optimistic estimates for measures having the weak dominance property.

Proposition 6.5. Let ϕ be a probability-based quality measure having the weak dominance property then $\phi^{oe} : \wp(\mathcal{G}) \to \mathbb{R}, S \to \phi(S \cap \mathcal{G}^+)$ is a tight optimistic estimates of ϕ.

Proof. Let $S \in \wp(\mathcal{G})$ and let $T \subseteq S$. We have $tpr(T) \leq tpr(S) = tpr(S \cap \mathcal{G}^+)$ and $fpr(T) \geq 0 = fpr(S \cap \mathcal{G}^+)$. Hence, using the weak dominance of ϕ we have: $(\forall T \subseteq S)\ \phi(T) \leq \phi(S \cap \mathcal{G}^+)$ making $\phi^{oe} : S \mapsto \phi(S \cap \mathcal{G}^+)$ a tight optimistic estimates since $S \cap \mathcal{G}^+ \subseteq S$. ▌

Note 6.10. There are other ways to build optimistic estimates. For instance, Morishita et al. [132] leverages the convexity of the χ^2 measure to obtain a tight optimistic estimates. □

We have seen in Table 6.2 many quality measures that are extent-based (since they are probability based). We give in Table 6.3 the optimistic estimates of the quality measures having a non-trivial tight optimistic estimates thanks to Proposition 6.5. Please note that the tight optimistic estimates is obtained for the measures having the weak dominance property by replacing the false positive rate by 0 and keeping the true positive rate the same.

6.4.1.2 Branch-and-Bound scheme

Algorithm 10 starts from the top subgroup $\mathcal{G} \in \mathbb{P}_{ext}$ and suppose that top already found is also $\mathcal{G}$ (Line 8). In some step of the algorithm, the call BRANCH-AND-BOUND(S, top_S) outputs the top possible subgroups w.r.t. ϕ in the sub-search space of subgroups enclosed in S if it exists, otherwise it does output top_S. To do so, it does rely on the notion of **optimistic estimates** ϕ^{oe} associated to ϕ [90] (see Definition 6.11): If the optimistic estimates of the quality on S is lower than the top-quality already found then the sub-search space is unpromising (Line 2). Otherwise, update the top-quality if S is better than the best already found subgroup (Line 3-4) and explore further subgroups (Line 5-6). Please note that when $\mathbb{P}$ is a pattern structure, one can

Algorithm 10: Brand-and-bound scheme for top-down subgroup enumeration

Input: $\mathbb{P}$ a pattern setup where $\mathcal{G} \in \mathbb{P}_{ext}$, ϕ an extent-based quality measure, ϕ^{oe} an optimistic estimates of ϕ.

Output: One element $top_S \in argmax_{S \in \mathbb{P}_{ext}} \phi(S)$

```
function BRANCH-AND-BOUND(S, top_S)
    if φ^oe(S) > φ(top_S) then
        if φ(S) > φ(top_S) then
            top_S ← S
        for all next refinements S' ⊊ S s.t. S' ∈ P_ext do
            top_S ← BRANCH-AND-BOUND(S', top_S)
    return top_S
print BRANCH-AND-BOUND(G, G)
```

visit only the *cotp* subgroups to find the top-quality subgroup as there exist at least one element in $argmax_{S \in \mathbb{P}_{ext}} \phi(S)$ that is relevant.

Note 6.11. This algorithm can be further extended to look for top-k subgroups w.r.t. a quality measure. However, and often, the top-k subgroups are very similar, i.e. they cover almost the same objects. Several attempts have been made in the literature to reduce this redundancy of the subgroup set: (1) limit the scope to relevant subgroups [82, 119], (2) diversification of the subgroups using a measure evaluating the redundancy of a subgroup set [160], or (3) diversification of the subgroup set using some similarity measure between subgroups [34, 145]. □

Measure	**Tight Optimistic Estimates**
Least contradiction [84]	y
Accuracy [84]	$\alpha \cdot y + (1-\alpha)$
Weighted Relative Accuracy [114]	$\alpha \cdot (1-\alpha) \cdot y$
Informedness [15]	y
Binomial Test [125]	$(1-\alpha) \cdot \sqrt{\alpha \cdot y}$
Klösgen$_\omega$ ($\omega \in [0,1]$) [101]	$(1-\alpha) \cdot (\alpha \cdot y)^\omega$
Linear correlation coefficient [154]	$\sqrt{\frac{(1-\alpha)\cdot y}{1-\alpha \cdot y}}$
Cohen's kappa (κ) [154]	$((2-2\cdot\alpha)\cdot y)/(1+(1-2\alpha)\cdot y)$
Cosine/G-Measure [84]	$\sqrt{y}$
m-estimate ($m \in (0,+\infty)$) [73]	$(\alpha \cdot [y + m/n])/(\alpha \cdot y + m/n)$
Discriminativity [30]	$n^2 \cdot \alpha \cdot (1-\alpha) \cdot y$
F_β score ($\beta \in [0,+\infty)$) [101]	$[(1+\beta^2)\cdot y]/[y+\beta^2]$
Gini Index [1]	$2 \cdot \alpha \cdot (1-\alpha) \cdot sup\left\{\frac{(1-\alpha)\cdot y}{1-\alpha\cdot y}, \frac{\alpha\cdot x}{1-(1-\alpha)\cdot x}\right\}$
Chi-Squared (χ^2) [1]	$n \cdot sup\left\{\frac{(1-\alpha)\cdot y}{1-\alpha\cdot y}, \frac{\alpha\cdot x}{1-(1-\alpha)\cdot x}\right\}$

Table 6.3: Tight optimistic estimates on some quality measures of Table 6.2 using the *true positive rate* y, the positive prevalence α and the size of the set of objects $|\mathcal{G}| = n$.

CHAPTER 7

ANYTIME DISCRIMINATIVE SUBGROUP DISCOVERY

This chapter can be seen as an application of the different notions we presented in the precedent chapters in this dissertation. We address here the problem of discovering subgroups that accurately discriminate one class label from the others in the particular case of the interval pattern language, i.e. axis-parallel hyper-rectangles. We have seen in Section 6.4 the different approaches existing in the literature to solve the aforementioned problem. In this chapter, we will present an anytime algorithm, namely REFINEANDMINE, tailored for discriminative interval patterns discovery in numerical data. It starts by mining interval patterns in a coarse discretization, followed by successive refinements yielding increasingly finer discretizations highlighting potentially new interesting patterns. Eventually, it performs an exhaustive search, if given enough time. Additionally, REFINEANDMINE gives two provable guarantees when interrupted. The first evaluates how close is the best found subgroup so far to the optimal one in the whole search space. The second measures how already found subgroups are diverse and cover well all the interesting regions in the dataset. These two guarantees makes algorithm REFINEANDMINE first of its kind when the task of discriminative subgroup discovery is considered. The following of the chapter is organized as follow:

- **Section 7.1** presents formally the problem we are investigating here in this chapter.
- **Section 7.2** introduces Algorithm REFINEANDMINE and shows that discretizations induce in fact complete sublattices of the complete lattice of interval patterns.
- **Section 7.3** presents the different guarantees provided by REFINEANDMINE.
- **Section 7.4** evaluates the performance of Algorithm REFINEANDMINE.
- **Section 7.5** provide a discussion on the possible future works to extend REFINEANDMINE.

The results presented in this chapter were introduced in paper [15] on which the writing of this chapter rely.

7.1 Problem Statement

In a general way, let $\mathbb{P} = (\mathcal{G}, (\mathcal{D}, \sqsubseteq), \delta)$ be a pattern structure over a finite set of objects $\mathcal{G}$ where $\mathcal{G}$ is partitioned into $\{\mathcal{G}^+, \mathcal{G}^-\}$. Let $\mathbb{P}_{ext}$ be the set of all subgroups and $\mathfrak{R} \subseteq \mathbb{P}_{ext}$ be the set of relevant subgroups towards the positive class. Let ϕ be a probabilistic based quality measure that has the **weak dominance property** (see Definition 6.9 and Table 6.2).

We want to design an **anytime enumeration algorithm** such that:

1. given enough time, outputs all relevant extents in $\mathfrak{R}$ (only **completeness** is required),
2. when interrupted, provides a guarantee bounding the difference of quality between the top-quality found subgroup and the top possible quality w.r.t. ϕ; and
3. outputs a second guarantee ensuring that the resulting patterns are diverse.

Formally, let $\mathbb{S}_i \subseteq \mathbb{P}_{ext}$ be the set of outputted solutions by the anytime algorithm at some step (or instant) i (at $i+1$ we have $\mathbb{S}_i \subseteq \mathbb{S}_{i+1}$). We want that (1) when i is big enough, $\mathbb{S}_i \supseteq \mathfrak{R}$. For (2) and (3), we define two metrics [1] to compare the results in $\mathbb{S}_i$ with the ones in $\mathfrak{R}$. The first metric, called $accuracy$ (eq. ACC), evaluates the difference between top subgroup quality ϕ in $\mathbb{S}_i$ and $\mathfrak{R}$ while the second metric, called $specificity$ (eq. SPE), evaluates how diverse and complete are subgroups in $\mathbb{S}_i$.

(ACC) $$accuracy_\phi(\mathbb{S}_i, \mathfrak{R}) = \sup_{A \in \mathfrak{R}} \phi(A) - \sup_{B \in \mathbb{S}_i} \phi(B)$$

(SPE) $$specificity(\mathbb{S}_i, \mathfrak{R}) = \sup_{A \in \mathfrak{R}} \inf_{B \in \mathbb{S}_i} \left(|A \,\Delta\, B| / |\mathcal{G}| \right)$$

The idea behind $specificity$ is that each extent A in $\mathfrak{R}$ is "approximated" by the most similar extent in $\mathbb{S}_i$; that is the set $B \in \mathbb{S}_i$ minimizing the *metric distance* $(A, B) \mapsto |A \,\Delta\, B| / |\mathcal{G}|$ in $\wp(\mathcal{G})$. The $specificity$ is then the highest possible distance (pessimistic). More formally it is the directed Hausdorff distance [99] from $\mathfrak{R}$ to $\mathbb{S}_i$. Note that $specificity(\mathbb{S}_i, \mathfrak{R}) = 0$ *is equivalent to* $\mathbb{S}_i \supseteq \mathfrak{R}$. Clearly, the lower these two metrics are, the closer we get to the desired output $\mathfrak{R}$. While $accuracy_\phi$ and $specificity$ can be *evaluated* when a complete exploration of $\mathfrak{R}$ is possible, our aim is to *bound* the two aforementioned measures independently from $\mathfrak{R}$ providing a *guarantee*. In other words, the anytime algorithm need to output additionally to $\mathbb{S}_i$, the two following measures $\overline{accuracy_\phi}(\mathbb{S}_i)$ and $\overline{specificity}(\mathbb{S}_i)$ s.t.

$$accuracy_\phi(\mathbb{S}_i, \mathfrak{R}) \leq \overline{accuracy_\phi}(\mathbb{S}_i)$$
$$specificity(\mathbb{S}_i, \mathfrak{R}) \leq \overline{specificity}(\mathbb{S}_i)$$

It is clear by definition that for all $i \in \mathbb{N}$ we have:

$$\begin{aligned} accuracy_\phi(\mathbb{S}_{i+1}, \mathfrak{R}) &\leq accuracy_\phi(\mathbb{S}_i, \mathfrak{R}) \\ specificity(\mathbb{S}_{i+1}, \mathfrak{R}) &\leq specificity(\mathbb{S}_i, \mathfrak{R}) \end{aligned}$$

[1] The metric names fall under the taxonomy of [172] for anytime algorithms.

We want also that the two bounds on both these measures decrease overtime providing better information on $\mathfrak{R}$ through $\mathbb{S}_i$, i.e.:

$$\begin{aligned} \overline{accuracy_\phi}(\mathbb{S}_{i+1}) &\leq \overline{accuracy_\phi}(\mathbb{S}_i) \\ \overline{specificity}(\mathbb{S}_{i+1}) &\leq \overline{specificity}(\mathbb{S}_i) \end{aligned}$$

7.2 Anytime Interval Pattern Mining

In this section, the pair $(\mathcal{G}, \mathcal{M})$ refers to a **finite numerical dataset** the considered pattern structure $\mathbb{P} = (\mathcal{G}, (\mathcal{D}, \sqsubseteq), \delta)$ refers to the **finite interval pattern structure** associated to $(\mathcal{G}, \mathcal{M})$ (see Definition 5.7). The set of objects $\mathcal{G}$ is partitioned into $\{\mathcal{G}^+, \mathcal{G}^-\}$ and the set of subgroups relevant to the positive class is denoted by $\mathfrak{R} \subseteq \mathbb{P}_{ext}$. Fig. 7.1 **(left)** depicts and example of a labeled numerical dataset $(\mathcal{G}, \mathcal{M})$. The algorithm that we will present later in this section for solving the problem in Section 7.1 for the particular case of interval pattern structures relies on the enumeration of a chain of discretizations from the coarsest to the finest. We start by defining the notion of **discretization** which, interestingly, induces a complete sublattices of the interval pattern language.

7.2.1 Discretizations as Complete Sublattices

A **discretization** of $\mathbb{R}$ is any *partition* of $\mathbb{R}$ using intervals. In particular, let $C = \{c_i\}_{1 \leq i \leq |C|} \subseteq \mathbb{R}$ be a finite set with $c_i < c_{i+1}$ for $i \in \{1, ..., |C| - 1\}$. Elements of C are called **cut points** or **cuts**. We associate to C a **finite discretization** denoted by $dr(C)$ and given by:

$$dr(C) = \{(-\infty, c_1)\} \cup \{[c_i, c_{i+1}) \mid i \in \{1, ..., |C| - 1]\}\} \cup \{[c_{|C|}, +\infty)\}$$

Generally speaking, let $C = (C_k)_{1 \leq k \leq |\mathcal{M}|} \in \wp(\mathbb{R})^{|\mathcal{M}|}$ representing **sets of cut points** associated to each dimension k (i.e. $C_k \subseteq \mathbb{R}$ finite $\forall k \in \{1, ..., |\mathcal{M}|\}$). The partition $dr(C)$ of $\mathbb{R}^{|\mathcal{M}|}$ is given by:

$$dr(C) = \left\{ \bigtimes_{k=1}^{|\mathcal{M}|} I_k \mid I_k \in dr(C_k) \right\}$$

Fig. 7.1 **(right)** depicts two discretizations. Discretizations are ordered using the natural order between partitions[2]. Moreover, cut-point sets are ordered by $\leq$ as follows:

$$C^1 \leq C^2 \equiv (\forall k \in \{1, ..., |\mathcal{M}|\}) C^1_k \subseteq C^2_k \text{ with } C^i = \left(C^i_k\right)_{1 \leq k \leq |\mathcal{M}|}$$

Clearly, if $C^1 \leq C^2$ then discretization $dr(C^1)$ **is coarser than** $dr(C^2)$.

[2]Let E be a set, a partition P_2 of E *is finer than* a partition P_1 (or P_1 *is coarser than* P_2) and we denote $P_1 \leq P_2$ if any subset in P_1 is a subset of a subset in P_2.

Let $C = (C_k)_{1 \leq k \leq |\mathcal{M}|}$ be the cut-points. Using the elementary hyper-rectangles (i.e. cells) in the discretization $dr(C)$, one can build a (finite) subset of descriptions $\mathcal{D}_C \subseteq \mathcal{D}$ which is the set of all possible descriptions that can be built using these cells. Formally:

$$\mathcal{D}_C := \left\{ \bigsqcap S \mid S \subseteq dr(C) \right\}$$

Note that $\top = \emptyset \in \mathcal{D}_C$ since $\bigsqcap \emptyset = \bigsqcup \mathcal{D} = \top$ by definition. Proposition 7.1 states that $(\mathcal{D}_C, \sqsubseteq)$ is a complete sublattice of $(\mathcal{D}, \sqsubseteq)$.

Proposition 7.1. For any finite set $C = (C_k)_{1 \leq k \leq |\mathcal{M}|} \in \wp(\mathbb{R})^{|\mathcal{M}|}$ of cut points, the description space $(\mathcal{D}_C, \sqsubseteq)$ with $\mathcal{D}_C := \left\{ \bigsqcap S \mid S \subseteq dr(C) \right\}$ is a **finite (complete) sublattice** of $(\mathcal{D}, \sqsubseteq)$ that is:

$$(\forall d_1, d_2 \in \mathcal{D}_C)\ d_1 \sqcup d_2 \in \mathcal{D}_C \textbf{ and } d_1 \sqcap d_2 \in \mathcal{D}_C$$

Moreover, if $C^1 \leq C^2$ are two cut-point sets, then $(\mathcal{D}_{C^1}, \sqsubseteq)$ is a (complete) sublattice of $(\mathcal{D}_{C^2}, \sqsubseteq)$.

Proof. By construction, $(\mathcal{D}_C, \sqsubseteq)$ is a **closure system** (preserve meet) on $(\mathcal{D}, \sqsubseteq)$. Indeed, let d_1 and d_2 be in $\mathcal{D}_C$ we have $\exists S_1, S_2 \subseteq dr(C)$ such that $d_1 = \bigsqcap S_1$ and $d_2 = \bigsqcap S_2$. Thus:

$$d_1 \sqcap d_2 = (\bigsqcap S_1) \sqcap (\bigsqcap S_2) = \bigsqcap \Big(S_1 \cup S_2 \Big) \in \mathcal{D}_C$$

since $S_1 \cup S_2 \subseteq dr(C)$.

Let us show now that $(\mathcal{D}_C, \sqsubseteq)$ is also a **kernel system** (preserve join) on $(\mathcal{D}, \sqsubseteq)$. We have $C = (C_k)_{1 \leq k \leq |\mathcal{M}|}$. Consider the following case of d_1 and d_2:

$$d_j = \bigtimes_{k=1}^{|\mathcal{M}|} I_k^j \text{ with } I_k^j = [a_k^j, b_k^j),\ j \in \{1,2\} \text{ and } a_k^j < b_k^j \in C_k$$

We have:

$$d_1 \sqcup d_2 = \bigtimes_{k=1}^{|\mathcal{M}|} [sup(a_k^1, a_k^2), inf(b_k^1, b_k^2))$$

Clearly, $d_1 \sqcup d_2 \in \mathcal{D}_C$, since the left bound $sup(a_k^1, a_k^2)$ and the right bound $inf(b_k^1, b_k^2)$ remains in C^k for all $k \in |\mathcal{M}|$. Note that if $sup(a_k^1, a_k^2) > inf(b_k^1, b_k^2)$ for at least one $k \in \{1..|\mathcal{M}|\}$, then $d_1 \sqcup d_2 = \top = \emptyset \in \mathcal{D}_C$. The two remaining cases of interval where we use $+\infty$ and $-\infty$ can be handled in, almost, the same way.

We conclude that $(\mathcal{D}_C, \sqsubseteq)$ is a finite (complete) sublattice of $(\mathcal{D}, \sqsubseteq)$ (since any finite lattice is by definition complete).

The second part of the proposition is straightforward since both posets $(\mathcal{D}_{C_1}, \sqsubseteq)$ and $(\mathcal{D}_{C_2}, \sqsubseteq)$ are complete sublattices of the same complete lattice $(\mathcal{D}, \sqsubseteq)$ and $\mathcal{D}_{C_1} \subseteq \mathcal{D}_{C_2}$ ($dr(C_1) \leq dr(C_2)$ since $C_1 \leq C_2$). ∎

	m_1	m_2	label
g_1	1	2	+
g_2	1	3	+
g_3	2	1	+
g_4	3	5	+
g_5	2	4	−
g_6	2	5	−
g_7	3	4	−
g_8	4	4	−

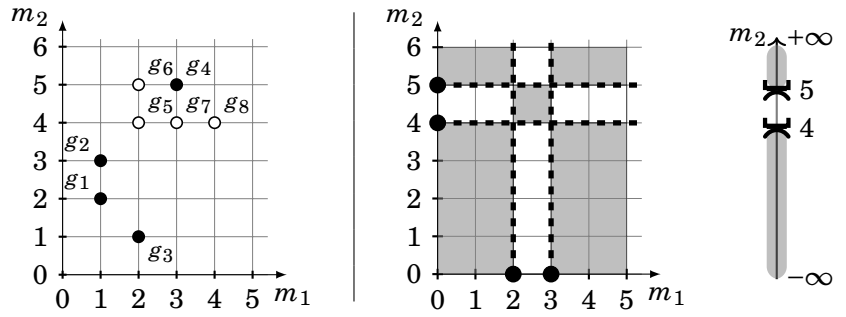


Figure 7.1: **From left to right:** A 2-dimensional numerical dataset, its representation on the Cartesian plane where black dots represent positive instances, discretization $dr((C_1,C_2))$ in $\mathbb{R}^2$ with $C_1 = \{2,3\}$ and $C_2 = \{4,5\}$ and **(right)** discretization $dr((C_2))$ in $\mathbb{R}$ representing the one associated with the relevant cuts of attribute m_2, i.e. $C_2 = C_2^{rel}$.

Note 7.1. Since according to Proposition 7.1, for any set of cutpoints C, the poset $(\mathcal{D}_C, \sqsubseteq)$ form a complete sublattice of the interval pattern language $(\mathcal{D}, \sqsubseteq)$, there exists a *meet-preserving kernel operator* $\psi_C : \mathcal{D} \to \mathcal{D}$ which fixpoints is $\psi[\mathcal{D}] = \mathcal{D}_C$ according to Theorem 2.8. This kernel operator is given by:

$$\psi_C : \mathcal{D} \to \mathcal{D}, d \mapsto \bigsqcup \{d_c \in \mathcal{D}_C \mid d_c \sqsubseteq d\}$$

Hence, one can create for each set of cutpoints C a projected pattern structure $\mathbb{P}_C := \psi_C(\mathbb{P})$ (see Definition 3.27):

$$\mathbb{P}_C := (\mathcal{G}, (\mathcal{D}_C, \sqsubseteq), \psi \circ \delta)$$

□

7.2.2 Finest Discretization for a Complete Enumeration of Relevant Extents

There exists cut points $C \subseteq \wp(\mathbb{R})^{|\mathcal{M}|}$ such that the space $(\mathcal{D}_C, \sqsubseteq)$ holds all relevant extents (i.e. $ext[\mathcal{D}_C] \supseteq \mathfrak{R}$). For instance, if we consider values appearing in the dataset as cut-points, i.e. $C = (m_k[\mathcal{G}])_{1 \leq k \leq |\mathcal{M}|}$, the description space $(\mathcal{D}_C, \sqsubseteq)$ holds all relevant extents since it does hold all extents. However, is there coarser discretization that holds all the relevant extents? The answer is affirmative. One can show that the only interesting cuts are those separating between positive and negative instances (called boundary cut-points by [63]). We call such cuts, **relevant cuts**. They are denoted by $C^{rel} = (C_k^{rel})_{1 \leq k \leq |\mathcal{M}|}$ and we have $ext[\mathcal{D}_{C^{rel}}] \supseteq \mathfrak{R}$. Formally, for each dimension k, a value $c \in m_k[\mathcal{G}]$ is a *relevant cut in* C_k^{rel} for attribute m_k iff:

$$\left(c \in m_k\left[\mathcal{G}^+\right] \text{ and } prev(c, m_k\left[\mathcal{G}\right]) \in m_k\left[\mathcal{G}^-\right]\right) \textbf{ or } \left(c \in m_k\left[\mathcal{G}^-\right] \text{ and } prev(c, m_k\left[\mathcal{G}\right]) \in m_k\left[\mathcal{G}^+\right]\right)$$

where for any $c \in \mathbb{R}$ and $A \subseteq \mathbb{R}$, we have:

$$next(c, A) = \inf\{a \in A \mid c < a\} \quad \textbf{and} \quad prev(c, A) = \sup\{a \in A \mid a < c\}$$

Finding *relevant cuts* C_k^{rel} is of the same complexity of sorting $m_k[\mathcal{G}]$ [63].

Example 7.1. In the dataset depicted in Fig. 7.1 **(left)**, relevant cuts are given by $C^{rel} = (\{2,3,4,5\},\{4,5\})$. Discretization $dr(C_2^{rel})$ is depicted in Fig. 7.1 (right). □

7.2.3 Anytime Enumeration of Relevant Extents

We design an *anytime* and *interruptible* algorithm dubbed REFINEANDMINE. This method, presented in Algorithm 11, relies on the enumeration of a chain of discretizations on the data space, from the coarsest to the finest. It begins by searching relevant cuts in pre-processing phase (line 2). Then, it builds a *coarse discretization* (line 3) containing a small set of relevant cut-points. Once the initial discretization built, *cotp* patterns are mined thanks to MININTCHANGE Algorithm (line 4) [104]. Then as long as the algorithm is not interrupted (or within the computational budget), we add new cut-points (line 6) building finer discretizations. For each added cut-point (line 8), only new interval patterns are searched for (mined descriptions d are new but their extents $ext(d)$ are not necessarily new) . That is *cotp* patterns which left or right bound is cut on the considered attribute $attr$ (i.e. $d.I_{attr} \in \{[cut,a),[cut,+\infty),[a,cut),(-\infty,cut) \mid a \in C^{cur}_{attr}\}$ with $d.I_{attr}$ is the $attr^{th}$ interval of d). This can be done by a slight modification of MININTCHANGE method (see Algorithm 7). REFINEANDMINE terminates when the set of relevant cuts is exhausted (i.e. $C^{cur} = C^{rel}$) ensuring a *complete* enumeration of relevant extents $\mathfrak{R}$.

The initial discretization (Line 3) can be done by various strategies (see [167]). A simple, yet efficient, choice is the equal frequency discretization with a fixed number of cuts. Other strategies can be used, e.g. [63]. Adding new cut-points (Line 6) can also be done in various ways. One strategy is to add a random relevant cut on a random attribute to build the next discretization. Section 7.3.3 proposes another more elaborated strategy that heuristically guide REFINEANDMINE to rapidly find good quality patterns (observed experimentally).

Algorithm 11: REFINEANDMINE

Input: $(\mathcal{G},\mathcal{M})$ a numerical datasets with $\{\mathcal{G}^+,\mathcal{G}^-\}$ partition of $\mathcal{G}$

1 **procedure REFINEANDMINE**()
2 Compute relevant cuts C^{rel}
3 Build an initial set of cut-points $C^{cur} \leq C^{rel}$
4 Mine *cotp* patterns in $\mathcal{D}_{C^{cur}}$ (and their extents) using MININTCHANGE
5 **while** $C^{cur} \neq C^{rel}$ *and within* ***computational budget*** **do**
6 Choose the next relevant cut $(attr,cut)$ with $cut \in C^{rel}_{attr} \setminus C^{cur}_{attr}$
7 Add the relevant cut cut to C^{cur}
8 Mine **new *cotp* pattern** in $(\mathcal{G},(\mathcal{D}_{C^{cur}},\sqsubseteq),\delta)$

Note 7.2. Algorithm REFINEANDMINE can be seen as an instance of a more general algorithm that traverse a chain of meet-preserving projected pattern structures. This is somehow quite similar to the methods proposed by Buzmakov et al. under the name of $\Sigma o\phi\iota\alpha$ [40, 42, 43]. □

7.3 Anytime Interval Pattern Mining with Guarantees

Algorithm REFINEANDMINE starts by mining patterns in a coarse discretization. It continues by mining more patterns in increasingly finer discretizations until the search space is totally explored (final complete lattice being $(\mathcal{D}_{C^{rel}}, \sqsubseteq)$). According to Proposition 7.1, the description spaces built on discretizations are complete sublattices of the total description space. This complete sublattices induce projected pattern structures as observed in Note 7.1.

For the sake of generality, in the following of this section $(\mathcal{D}, \sqsubseteq)$ denotes a *complete lattice*, and for all $i \in \mathbb{N}^*$, $(\mathcal{D}_i, \sqsubseteq)$ denotes *complete sublattices* of $(\mathcal{D}, \sqsubseteq)$ such that $\mathcal{D}_i \subseteq \mathcal{D}_{i+1} \subseteq \mathcal{D}$. For instance, in REFINEANDMINE, the total complete lattice is $(\mathcal{D}_{C^{rel}}, \sqsubseteq)$ while the $(\mathcal{D}_i, \sqsubseteq)$ are $(\mathcal{D}_{C^{cur}}, \sqsubseteq)$ at each step. Before giving the formulas of $\overline{accuracy_\phi}(\mathbb{S}_i)$ and $\overline{specificity}(\mathbb{S}_i)$, we give some necessary definitions and underlying properties. At the end of this section, we instanciates the different general results on the particular case of REFINEANDMINE for interval patterns.

7.3.1 Approximating Descriptions in a Complete Sublattice

7.3.1.1 Upper and lower approximations of patterns

We start by approximating each pattern in $\mathcal{D}$ using two patterns in $\mathcal{D}_i$. Before presenting the formal definition, let us consider the following example.

Example 7.2. Consider for instance Fig. 7.2 where $\mathcal{D}$ is the space of interval patterns in $\mathbb{R}^2$ while $\mathcal{D}_C$ is the space containing only rectangles that can be built over discretization $dr(C)$ with $C = (\{1,4,6,8\},\{1,3,5,6\})$. Since the hatched rectangle $d = [3,7] \times [2,5.5] \in \mathcal{D}$ does not belong to $\mathcal{D}_C$, two descriptions in $\mathcal{D}_C$ can be used to encapsulate it. The first one, depicted by a gray rectangle, is called the *upper approximation* of d. It is given by the smallest rectangle in $\mathcal{D}_C$ enclosing d. Dually, the second approximation represented as a black rectangle and coined *lower approximation* of d, is given by the greatest rectangle in $\mathcal{D}_C$ enclosed by d. □

The two denominations of lower and upper approximation used in Example 7.2 come from Rough Set Theory [140] where lower and upper approximations form together a *rough set* and try to capture the undefined rectangle $d \in \mathcal{D} \setminus \mathcal{D}_C$. Definition 7.1 formalizes these two approximations in the general case.

Definition 7.1. The upper approximation mapping $\overline{\psi_i}$ and lower approximation mapping $\underline{\psi_i}$ are the mappings defined as follows:

$$\overline{\psi_i} : \mathcal{D} \to \mathcal{D}_i, d \mapsto \bigsqcup \{c \in \mathcal{D}_i \mid c \sqsubseteq d\} \qquad \underline{\psi_i} : \mathcal{D} \to \mathcal{D}_i, d \mapsto \bigsqcap \{c \in \mathcal{D}_i \mid d \sqsubseteq c\}$$

The existence of these two mappings is ensured by the fact that $(\mathcal{D}_i, \sqsubseteq)$ is a complete sublattice of $(\mathcal{D}, \sqsubseteq)$. Proposition 7.2 states an important property.

Proposition 7.2. We have $(\forall d \in \mathcal{D})\ \overline{\psi_i}(d) \sqsubseteq d \sqsubseteq \underline{\psi_i}(d)$.

Proof. Proposition 7.2 is a small result from Theorem 2.8. Since $(\mathcal{D}_i, \sqsubseteq)$ is a complete sublattice of $(\mathcal{D}, \sqsubseteq)$, then the mappings $\underline{\psi_i^*}$ and $\overline{\psi_i^*}$ defined below

$$\underline{\psi_i^*}(d) : \mathcal{D} \to \mathcal{D}, d \mapsto \underline{\psi_i}(d) \quad \textbf{and} \quad \overline{\psi_i^*}(d) : \mathcal{D} \to \mathcal{D}, d \mapsto \overline{\psi_i}(d)$$

are respectively *join-preserving closure* (thus extensive $d \sqsubseteq \underline{\psi_i^*}(d)$) and *meet-preserving kernel* (thus contractive $\overline{\psi_i^*}(d) \sqsubseteq d$) operators on $(\mathcal{D}, \sqsubseteq)$ with $\underline{\psi_i^*}[\mathcal{D}] = \left\{\underline{\psi_i^*}(d) \mid d \in \mathcal{D}_i\right\} = \mathcal{D}_i$ and $\overline{\psi_i^*}[\mathcal{D}] = \mathcal{D}_i$. Recalling that *ext* is order-reversing concludes the proof. ∎

Another useful Lemma is presented here.

Lemma 7.1. The richer $(\mathcal{D}_i, \sqsubseteq)$ is, the more constraining are the surrounding approximations. Formally:

$$(\forall d \in \mathcal{D})\ \overline{\psi_i}(d) \sqsubseteq \overline{\psi_{i+1}}(d) \sqsubseteq d \sqsubseteq \underline{\psi_{i+1}}(d) \sqsubseteq \underline{\psi_i}(d)$$

Generally speaking: $\overline{\psi_i} \circ \overline{\psi_{i+1}} = \overline{\psi_i}$ **and** $\underline{\psi_i} \circ \underline{\psi_{i+1}} = \underline{\psi_i}$.

Proof. Without loss of generality, let $i = 1$. Let us show before that $\overline{\psi_1}\big(\overline{\psi_2}(d)\big) = \overline{\psi_1}(d)$ and $\underline{\psi_1}\big(\underline{\psi_2}(d)\big) = \underline{\psi_1}(d)$ for all $d \in \mathcal{D}$.
Since $\mathcal{D}_1 \subseteq \mathcal{D}_2$ then for all $x \in \mathcal{D}_1$ and for all $d \in \mathcal{D}$ we have (since $\overline{\psi_2}$ and $\underline{\psi_2}$ come respectively from kernel and closure operators):

$$x \sqsubseteq \overline{\psi_2}(d) \Leftrightarrow x \sqsubseteq d \quad \textbf{and} \quad \underline{\psi_2}(d) \sqsubseteq x \Leftrightarrow d \sqsubseteq x$$

Thus:

$$\overline{\psi_1}\big(\overline{\psi_2}(d)\big) = \bigsqcup\{x \in \mathcal{D}_1 \mid x \sqsubseteq \overline{\psi_2}(d)\} = \bigsqcup\{x \in \mathcal{D}_1 \mid x \sqsubseteq d\} = \overline{\psi_1}(d)$$
$$\underline{\psi_1}\big(\underline{\psi_2}(d)\big) = \bigsqcap\{x \in \mathcal{D}_1 \mid \underline{\psi_2}(d) \sqsubseteq x\} = \bigsqcap\{x \in \mathcal{D}_1 \mid d \sqsubseteq x\} = \underline{\psi_1}(d)$$

The first part of the proposition is a straight-forward corollary of the first properties. Indeed, since $\overline{\psi_2}$ and $\underline{\psi_2}$ are respectively *contractive* and *extensive*, we conclude:

$$\overline{\psi_1}(d) = \overline{\psi_1}\big(\overline{\psi_2}(d)\big) \sqsubseteq \overline{\psi_2}(d) \quad \textbf{and} \quad \underline{\psi_2}(d) \sqsubseteq \underline{\psi_1}\big(\underline{\psi_2}(d)\big) = \underline{\psi_1}(d)$$

Thus: $\overline{\psi_1}(d) \sqsubseteq \overline{\psi_2}(d) \sqsubseteq d \sqsubseteq \underline{\psi_2}(d) \sqsubseteq \underline{\psi_1}(d)$. ∎

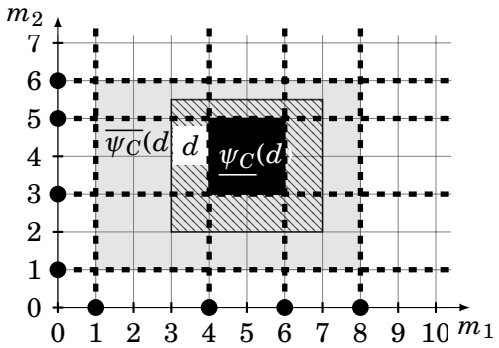


Figure 7.2: Description $d = [3,7] \times [2,5.5]$ in $\mathcal{D}$ (hatched) and $C = (\{1,4,6,8\},\{1,3,5,6\})$. Upper approximation of d in $\mathcal{D}_C$ is $\overline{\psi_C}(d) = [1,8) \times [1,6)$ (gray rectangle) while lower approximation of d is $\underline{\psi_C}(d) = [4,6) \times [3,5)$ (black rectangle). $\underline{\psi_C}(d)$ represents also the $core_C$ of $\overline{\psi_C}(d)$ in $\mathcal{D}_C$.

7.3.1.2 Encapsulating patterns using their upper-approximations.

We want to encapsulate any description $d \in \mathcal{D}$ by knowing only its upper-approximation in $\mathcal{D}_i$. Formally, we want some function $f : \mathcal{D}_i \to \mathcal{D}_i$ such that $(\forall d \in \mathcal{D}) \overline{\psi_i}(d) \sqsubseteq d \sqsubseteq f(\overline{\psi_i}(d))$. Definition 7.2 defines the notion of *core* mapping and Proposition 7.3 shows that such a mapping *core* is the tightest function f (w.r.t. $\sqsubseteq$) that we want to build.

Definition 7.2. The mapping $core_i$ associated to $\mathcal{D}_i$ is given by:

$$core_i : \mathcal{D}_i \to \mathcal{D}_i, c \mapsto core(c) = \underline{\psi_i}\left(\bigsqcup \left\{d \in \mathcal{D} \mid \overline{\psi_i}(d) = c\right\}\right)$$

Proposition 7.3. For any mapping $f : \mathcal{D}_i \to \mathcal{D}_i$ we have:

$$(\forall d \in \mathcal{D})\ d \sqsubseteq f(\overline{\psi_i}(d)) \iff (\forall c \in \mathcal{D}_i)\ core_i(c) \sqsubseteq f(c)$$

Proof. Let $f : \mathcal{D}_i \to \mathcal{D}_i$ be a function. We want to show the following property:

$$(\forall d \in \mathcal{D})\ d \sqsubseteq f(\overline{\psi_i}(d)) \iff (\forall c \in \mathcal{D}_i)\ core_i(c) \sqsubseteq f(c)$$

We start by implication ($\Leftarrow$). Let $d \in \mathcal{D}$, since from the hypothesis $core_i(\overline{\psi_i}(d)) \sqsubseteq f(\overline{\psi_i}(d))$ and since $d \sqsubseteq core_i(\overline{\psi_i}(d))$ (Proposition 7.4), we conclude that $d \sqsubseteq f(\overline{\psi_i}(d))$.

It remains to show the other implication ($\Leftarrow$). Let $c \in \mathcal{D}_i$ and let $S_c := \{x \in \mathcal{D} \mid \overline{\psi_i}(x) = c\}$. Since, from the hypothesis, $(\forall x \in S_c)\ x \sqsubseteq f(\overline{\psi_i}(x)) = f(c)$ then ($f(c)$ is an upper bound of S_c and the join is the smallest upper bound by definition, i.e.: $\bigsqcup S_c \sqsubseteq f(c)$. Since $f(c) \in \mathcal{D}_i$ we have $\underline{\psi_i}(f(c)) = f(c)$. We obtain by monotonicity of $\underline{\psi_i}$: $core_i(c) = \underline{\psi_i}(\bigsqcup S_c) \sqsubseteq \underline{\psi_i}(f(c)) = f(c)$ This concludes the demonstration. ∎

Proposition 7.4 shows that the core of the upper approximation is always less restrictive than the lower approximation.

Proposition 7.4. We have:

$$(\forall d \in \mathcal{D})\ \overline{\psi_i}(d) \sqsubseteq d \sqsubseteq \underline{\psi_i}(d) \sqsubseteq core_i(\overline{\psi_i}(d))$$

Proof. We have from Proposition 7.2 that $(\forall d \in \mathcal{D})\ \overline{\psi_i}(d) \sqsubseteq d \sqsubseteq \underline{\psi_i}(d)$. Let us show now that:

$$(\forall d \in \mathcal{D})\ \underline{\psi_i}(d) \sqsubseteq core_i(\overline{\psi_i}(d)) = \underline{\psi_i}\left(\bigsqcup \left\{x \in \mathcal{D} \mid \overline{\psi_i}(x) = \overline{\psi_i}(d)\right\}\right)$$

Clearly, we have $d \in \left\{x \in \mathcal{D} \mid \overline{\psi_i}(x) = \overline{\psi_i}(d)\right\}$. We obtain hence that

$$d \sqsubseteq \bigsqcup \left\{x \in \mathcal{D} \mid \overline{\psi_i}(x) = \overline{\psi_i}(d)\right\}$$

We conclude the proof using *monotonicity* of $\underline{\psi_i}$. ∎

Note that, while the $core$ operator definition depends clearly on the complete lattice $(\mathcal{D}, \sqsubseteq)$, its computation should be done independently from $(\mathcal{D}, \sqsubseteq)$.

Another important property of the core is presented in the following Lemma.

Lemma 7.2. $\forall d \in \mathcal{D} : core_{i+1}(\overline{\psi_{i+1}}(d)) \sqsubseteq core_i(\overline{\psi_i}(d))$. That is, the core of the upper approximation is less restrictive in richer spaces.

Proof. Without loss of generality, let $i = 1$. Let $d \in \mathcal{D}$, we need to show that $core_2(\overline{\psi_2}(d)) \sqsubseteq core_1(\overline{\psi_1}(d))$. According to Lemma 7.1 we have $\overline{\psi_1} \circ \overline{\psi_2} = \overline{\psi_1}$, thus:

$$\begin{array}{rcl} \left\{x \in \mathcal{D} \mid \overline{\psi_2}(x) = \overline{\psi_2}(d)\right\} & \subseteq & \left\{x \in \mathcal{D} \mid \overline{\psi_1}(x) = \overline{\psi_1}(d)\right\} \Longrightarrow \\ \bigsqcup \left\{x \in \mathcal{D} \mid \overline{\psi_2}(x) = \overline{\psi_2}(d)\right\} & \sqsubseteq & \bigsqcup \left\{x \in \mathcal{D} \mid \overline{\psi_1}(x) = \overline{\psi_1}(d)\right\} \end{array}$$

Since $\underline{\psi_2}$ is monotonic, we obtain:

$$\underline{\psi_2}\left(\bigsqcup \left\{x \in \mathcal{D} \mid \overline{\psi_2}(x) = \overline{\psi_2}(d)\right\}\right) \sqsubseteq \underline{\psi_2}\left(\bigsqcup \left\{x \in \mathcal{D} \mid \overline{\psi_1}(x) = \overline{\psi_1}(d)\right\}\right)$$

Since $\underline{\psi_1}$ is extensive and $\underline{\psi_1} \circ \underline{\psi_2} = \underline{\psi_1}$, we obtain:

$$\underline{\psi_2}\left(\bigsqcup \left\{x \in \mathcal{D} \mid \overline{\psi_2}(x) = \overline{\psi_2}(d)\right\}\right) \sqsubseteq \underline{\psi_1}\left(\bigsqcup \left\{x \in \mathcal{D} \mid \overline{\psi_1}(x) = \overline{\psi_1}(d)\right\}\right)$$

We conclude that:

$$core_2(\overline{\psi_2}(d)) \sqsubseteq core_1(\overline{\psi_1}(d))$$

This ends the demonstration. ∎

7.3.2 Bounding accuracy and specificity Metrics

We consider now the pattern structure $\mathbb{P} = (\mathcal{G}, (\mathcal{D}, \sqsubseteq), \delta)$ associated to the complete lattice $(\mathcal{D}, \sqsubseteq)$ where the set of $\mathcal{G}$ is partitioned into $\{\mathcal{G}^+, \mathcal{G}^-\}$. Since $(\mathcal{D}_i, \sqsubseteq)$ are complete sublattices of $(\mathcal{D}, \sqsubseteq)$. One can create for each $i \in \mathbb{N}^*$, the projected[3] pattern structure $\mathbb{P}_i := \overline{\psi_i}(\mathbb{P})$ given by:

$$\mathbb{P}_i := (\mathcal{G}, (\mathcal{D}, \sqsubseteq), \overline{\psi_i} \circ \delta)$$

Moreover, the associated extent ext_i and intent int_i operators of $\mathbb{P}_i$ are given by (see Proposition 3.11 and Theorem 3.9).

$$ext_i : \mathcal{D}_i \to \wp(\mathcal{G}), d \mapsto ext(d) \quad \textbf{and} \quad int_i : \wp(\mathcal{G}) \to \mathcal{D}_i, A \mapsto \overline{\psi_i}(int(A))$$

Please note that, thanks to the fact that ext is order-reversing and Proposition 7.2, we have:

$$(\forall d \in \mathcal{D})\ ext(\underline{\psi_i}(d)) \subseteq ext(d) \subseteq ext(\overline{\psi_i}(d))$$

Following REFINEANDMINE, we consider that at the i^{th} step the algorithm visiting the projected pattern structures outputs extents $\mathbb{S}_i$ which corresponds to the *cotp* extents of $\mathbb{P}_i$, i.e.:

$$\mathbb{S}_i := ext \circ int_i[\wp(\mathcal{G}^+)] = ext[\overline{\psi_i}[int[\wp(\mathcal{G}^+)]]]$$

The next sections provide the bounds $\overline{accuracy_\phi}$ and $\overline{specificity}$ given a probability based quality measure ϕ that has the weak dominance property (see Definition 6.8 and Definition 6.9). Please read Note 6.4 for the different notations we use to manipulate a probabiliy based quality measure ϕ.

7.3.2.1 Bounding the accuracy metric

Before giving the general formula of $\overline{accuracy_\phi}$, we state below the following Lemma.

Lemma 7.3. We have:

$$(\forall d \in \mathcal{D})\ \phi(d) \leq \phi\Big(fpr\Big(core_i\big(\overline{\psi_i}(d)\big)\Big), tpr\Big(\overline{\psi_i}(d)\Big)\Big)$$

Proof. By application of Proposition 7.3 we have with $c = \overline{\psi_i}(d) \in \mathcal{D}_i$: $c \sqsubseteq d \sqsubseteq core_i(c)$. Thus, since ext operator is order-reversing we obtain $ext(core_i(c)) \subseteq ext(d) \subseteq ext(c)$ Thus, since tpr and fpr increase with the extent. We conclude that $tpr(core_i(c)) \leq tpr(d) \leq tpr(c)$ and $fpr(core_i(c)) \leq fpr(d) \leq fpr(c)$. Since ϕ has the *weak dominance property* we conclude that: $\phi(d) \leq \phi(fpr(core_i(c)), tpr(c))$. This ends the proof. ∎

Theorem 7.1. The $accuracy$ metric is bounded by:

$$\overline{accuracy_\phi}(\mathbb{S}_i) \quad = \quad \sup_{c \in int_i[\mathbb{S}_i]} \left[\phi\big(fpr\big(core_i(c)\big), tpr\big(c\big)\big) - \phi(fpr(c), tpr(c))\right]$$

[3] There is a slight misuse of the term projection with the mapping $\overline{\psi_i}$ since its codomain is $\overline{\psi_i}[\mathcal{D}] = \mathcal{D}_i$ rather than $\mathcal{D}$. If the codomain is $\mathcal{D}$, the mapping is a kernel operator.

Proof. According to lemma 7.3 and by considering $d \in int[\wp(\mathcal{G}^+)]$, a *cotp* pattern we have:

$$\phi(d) \quad \leq \quad \phi\Big(fpr\Big(core_1(\overline{\psi_1}(d))\Big), tpr\Big(\overline{\psi_1}(d)\Big)\Big)$$

Since $\overline{\psi_1}(d)$ is a *cotp* pattern induced by $(\mathcal{D}_1, \sqsubseteq)$, we have:

$$\sup_{int[\wp(\mathcal{G}^+)]} \phi(d) \leq \sup_{c \in \overline{\psi_1}[int[\wp(\mathcal{G}^+)]]} \phi\Big(fpr\Big(core_1(c)\Big), tpr\Big(c\Big)\Big)$$

The set $\overline{\psi_1}[int[\wp(\mathcal{G}^+)]]$ represent sthe set of *cotp* patterns in $\mathcal{D}_1$, that is $int_1[\mathbb{S}_1]$ ($\mathbb{S}_1$ represents the set of *cotp* extents induced by $\mathcal{D}_1$), we obtain:

$$\sup_{d \in int[\wp(\mathcal{G}^+)]} \phi(d) \leq \sup_{c \in int_1[\mathbb{S}_1]} \phi\Big(tpr\Big(c\Big), fpr\Big(core_1(c)\Big)\Big)$$

Thus, since the left quantity is the same as $\sup_{A \in \mathcal{R}} \phi(A)$, we subtract from both side the quantity $\sup_{B \in \mathbb{S}_1} \phi(B) = \sup_{c \in int_1[\mathbb{S}_1]} \phi(fpr(c), tpr(c))$. Hence:

$$\sup_{A \in \mathcal{R}} \phi(A) - \sup_{B \in \mathbb{S}_1} \phi(B) \quad \leq \quad \sup_{c \in int_1[\mathbb{S}_1]} \Big[\phi\big(fpr\big(core_1(c)\big), tpr\big(c\big)\big) - \phi(c)\Big]$$

That is: $\overline{accuracy_\phi}(\mathbb{S}_1) = \sup_{c \in int_1[\mathbb{S}_1]} \Big[\phi\big(fpr\big(core_1(c)\big), tpr\big(c\big)\big) - \phi(fpr(c), tpr(c))\Big]$. ∎

Proposition 7.5. We have $\overline{accuracy_\phi}(\mathbb{S}_{i+1}) \leq \overline{accuracy_\phi}(\mathbb{S}_i)$.

Proof. From Lemma 7.1 and Lemma 7.2 we have for all $d \in \mathcal{D}$:

$$\overline{\psi_1}(d) \sqsubseteq \overline{\psi_2}(d) \sqsubseteq d \sqsubseteq core_2(\overline{\psi_2}(d)) \sqsubseteq core_1(\overline{\psi_1}(d))$$

Thus, since ϕ has the weak dominance property, we have for all $d \in \mathcal{D}$:

$$\phi\Big(fpr\Big(core_2(\overline{\psi_2}(d))\Big), tpr\Big(\overline{\psi_2}(d)\Big)\Big) \leq \phi\Big(fpr\Big(core_1(\overline{\psi_1}(d))\Big), tpr\Big(\overline{\psi_1}(d)\Big)\Big)$$

Particularly, for $c \in \mathcal{D}_2$ (c is a fixpoint for $\overline{\psi_2}$) we have:

$$\phi\Big(fpr\Big(core_2(c)\Big), tpr\Big(c\Big)\Big) \leq \phi\Big(fpr\Big(core_1(\overline{\psi_1}(c))\Big), tpr\Big(\overline{\psi_1}(c)\Big)\Big)$$

We conclude that:

$$\sup_{c \in int_2[\mathbb{S}_2]} \phi\big(fpr\big(core_2(c)\big), tpr\big(c\big)\big) \quad \leq \quad \sup_{c \in int_1[\mathbb{S}_1]} \phi\big(fpr\big(core_1(c)\big), tpr\big(c\big)\big)$$

Moreover, since $\mathcal{D}_1 \subseteq \mathcal{D}_2$, we have $-\sup_{c \in \mathcal{D}_2} \phi(c) \leq -\sup_{c \in \mathcal{D}_1} \phi(c)$. Therefore, we obtain that $-\sup_{c \in int_2[\mathbb{S}_2]} \phi(c) \leq -\sup_{c \in int_1[\mathbb{S}_1]} \phi(c)$. We conclude $\overline{accuracy_\phi}(\mathbb{S}_2) \leq \overline{accuracy_\phi}(\mathbb{S}_1)$ ∎

7.3.2.2 Bounding the specificity metric

Before giving the general formula of $specificity$, we state below the following Lemma.

Lemma 7.4. We have:

$$(\forall d \in int[\wp(\mathcal{G}^+)]) \inf_{c \in int_i[\mathbb{S}_i]} \left(\frac{|ext(d) \Delta ext(c)|}{|\mathcal{G}|} \right) \leq \frac{|ext(\overline{\psi_i}(d))| - |ext(core_i^+(\overline{\psi_i}(d)))|}{2 \cdot |\mathcal{G}|}$$

where $core_i^+ : c \mapsto int_i\left(ext(core_i(c)) \cap \mathcal{G}^+\right)$ the closure on the positives of $core_i(c)$.

Proof. Since the upper approximations (which is a *cotp* in $\mathbb{P}_i$ and thus is in $int_i[\mathbb{S}_i]$) and the closure on the positives of the core (it is in $int_i[\mathbb{S}_i]$) are already good approximations for d (but not necessarily the bests), let $u = \overline{\psi_i}(d)$, we have $ext(d) \subseteq ext(u)$. On the other hand, since $d \sqsubseteq core_i(u) \sqsubseteq core_i^+(u)$ (Proposition 7.3), we have $ext(core_i^+(u)) \subseteq ext(d)$. Thus, we have $|ext(d) \Delta ext(u)| = |ext(u)| - |ext(d)|$ and in the other hand $|ext(d) \Delta ext(core_i^+(u))| = |ext(d)| - |ext(core_i^+(u))|$. We obtain:

$$\begin{aligned} \inf_{c \in int_i[\mathbb{S}_i]} |ext(d) \Delta ext(c)| &\leq |ext(u)| - |ext(d)| \\ \inf_{c \in int_i[\mathbb{S}_i]} |ext(d) \Delta ext(c)| &\leq |ext(d)| - |ext(core_i^+(u))| \end{aligned}$$

We conclude:

$$2 \times \inf_{c \in int_i[\mathbb{S}_i]} |ext(d) \Delta ext(c)| \leq |ext(u)| - |ext(core_i^+(u))|$$

Since $|\mathcal{G}|$ is a constant, we obtain (with $u = \overline{\psi_i}(d)$):

$$\inf_{c \in int_i[\mathbb{S}_i]} \left(\frac{|ext(d) \Delta ext(c)|}{|\mathcal{G}|} \right) \leq \frac{|ext(u)| - |ext(core_i^+(u))|}{2 \cdot |\mathcal{G}|}$$

This ends the demonstration. ∎

Theorem 7.2. The $specificity$ metric is bounded by:

$$\overline{specificity}(\mathbb{S}_i) = \sup_{c \in int_i[\mathbb{S}_i]} \left(\frac{|ext(c)| - |ext(core_i^+(c))|}{2 \cdot |\mathcal{G}|} \right)$$

Proof. Without loss of generality, let $i = 1$. According to Lemma 7.4, we have for a relevant (thus *cotp*) extent $A \in \mathcal{R}$ ($int_1(A) \in int_1[\wp(\mathcal{G}^+)]$] and thus $int_1(A) \in int_1[\mathbb{S}_1]$):

$$\inf_{B \in S_1} \left(\frac{|A \,\Delta B|}{|\mathcal{G}|} \right) \leq \frac{|ext(int_1(A))| - |ext(core_1^+(int_1(A)))|}{2 \cdot |\mathcal{G}|}$$

We conclude thus:

$$\sup_{A \in \mathcal{R}} \inf_{B \in \mathbb{S}_1} \left(\frac{|A \,\Delta B|}{|\mathcal{G}|} \right) \quad \leq \quad \sup_{c \in int_1[\mathbb{S}_1]} \frac{|ext(c)| - |ext(core_1^+(c))|}{2 \cdot |\mathcal{G}|}$$

That is to say:

$$\overline{specificity}(\mathbb{S}_1) \quad = \quad \sup_{c \in int_1[\mathbb{S}_1]} \frac{|ext(c)| - |ext(core_1^+(c))|}{2 \cdot |\mathcal{G}|}$$

This ends the demonstration. ∎

Proposition 7.6. We have $\overline{specificity}(\mathbb{S}_{i+1}) \leq \overline{specificity}(\mathbb{S}_i)$.

Proof. Let us show that $\overline{specificity}$ is *order reversing*. From Lemma 7.1 and Lemma 7.2 we have for all $d \in \mathcal{D}$:

$$\overline{\psi_1}(d) \sqsubseteq \overline{\psi_2}(d) \sqsubseteq d \sqsubseteq core_2(\overline{\psi_2}(d)) \sqsubseteq core_1(\overline{\psi_1}(d))$$

We can conclude directly using Lemma 7.4 (recall that *ext* is an order reversing) that for all $d \in \mathcal{D}$:

$$\frac{|ext(\overline{\psi_2}(d))| - |ext(core_2(\overline{\psi_2}(d)))|}{2 \cdot |\mathcal{G}|} \quad \leq \quad \frac{|ext(\overline{\psi_1}(d))| - |ext(core_1(\overline{\psi_1}(d)))|}{2 \cdot |\mathcal{G}|}$$

Particularity, for $c \in \mathcal{D}_2$ (c is a fix-point for $\overline{\psi_2}$) we obtain:

$$\frac{|ext(c)| - |ext(core_2(c))|}{2 \cdot |\mathcal{G}|} \quad \leq \quad \frac{|ext(\overline{\psi_1}(c))| - |ext(core_1(\overline{\psi_1}(c)))|}{2 \cdot |\mathcal{G}|}$$

We conclude that:

$$\sup_{c \in int_2[\mathbb{S}_2]} \frac{|ext(c)| - |ext(core_2^+(c))|}{2 \cdot |\mathcal{G}|} \quad \leq \quad \sup_{c \in int_1[\mathbb{S}_1]} \frac{|ext(c)| - |ext(core_1^+(c))|}{2 \cdot |\mathcal{G}|}$$

In other word:

$$\overline{specificity}(\mathbb{S}_2) \quad \leq \quad \overline{specificity}(\mathbb{S}_1)$$

This conclude the demonstration. ∎

7.3.3 Instanciation on REFINEANDMINE

We show below how the different steps of the method REFINEANDMINE (see Algorithm 11) should be updated in order to compute the two bounds $\overline{accuracy}$ and $\overline{specificity}$. For the sake of simplicity, we explain here a naive approach as we intend to provide an overview of the algorithm. Note that here, $core$ (resp. $core^+$) refers to $core_{C^{cur}}$ (resp. $core^+_{C^{cur}}$).

7.3.3.1 Computing the core

In each step and for cut-points $C = (C_k) \subseteq \wp(\mathbb{R})^{|\mathcal{M}|}$, the finite lattice $(\mathcal{D}_C, \sqsubseteq)$ is a sublattice of the finest finite lattice $(\mathcal{D}_{C^{rel}}, \sqsubseteq)$ (since $C \leq C^{rel}$). Thereby, the $core$ is computed according to this latter as follows: Let $d \in \mathcal{D}_C$ with $d.I_k = [a_k, b_k)$ for all $k \in \{1,...,|\mathcal{M}|\}$. The left (resp. right) bound of $core_C(d).I_k$ for any k is equal to $next(a_k, C_k)$ (resp. $prev(b_k, C_k)$) if $next(a_k, C^{rel}_k) \notin C_k$ (resp. $prev(b_k, C^{rel}_k) \notin C_k$). Otherwise, it is equal to a_k (resp. b_k).

Example 7.3. Consider the step $\mathcal{C} = (\{2,3\},\{4,5\})$ in REFINEANDMINE (its associated discretization is depicted in Fig. 7.1 (left)) and recall that the relevant cuts set is $\mathcal{C}^{rel} = (\{2,3,4,5\},\{4,5\})$. The core of the bottom pattern $\bot = \mathbb{R}^2$ at this step is $core_{C^{cur}}(\bot) = (-\infty, 3) \times \mathbb{R}$. Indeed, there is three descriptions in $\mathcal{D}_{C^{rel}}$ which upper approximation is $\bot$, namely $\bot$, $c_1 = (-\infty, 4) \times \mathbb{R}$ and $c_2 = (-\infty, 5) \times \mathbb{R}$. Their lower approximations are respectively $\bot$, $(-\infty, 3) \times \mathbb{R}$ and $(-\infty, 3) \times \mathbb{R}$. The join (intersection) of these three descriptions is then $core_{C^{cur}}(\bot) = (-\infty, 3) \times (-\infty, +\infty)$. □

7.3.3.2 Computing the initial bounds (line 4)

As MININTCHANGE enumerates all *cotp* patterns $d \in \mathcal{D}_{C^{cur}}$, REFINEANDMINE stores in a key-value structure (i.e. map) called `BoundPerPosExt` the following entries:

$$ext(d) \cap \mathcal{G}^+ : \left(\phi(d), \phi\big(fpr\big(core(d)\big), tpr\big(d\big)\big), \frac{|ext(d)| - |ext(core^+(d))|}{2 \cdot |\mathcal{G}|} \right)$$

The *error-bounds* $\overline{accuracy_\phi}$ and $\overline{specificity}$ are then computed at the end by a single pass on the entries of `BoundPerPosExt` using Theorems 7.1 and 7.2.

7.3.3.3 Updating the bounds after adding a new cut-point (line 8)

In order to compute the new *error-bounds* $\overline{accuracy_\phi}$ and $\overline{specificity}$ which decrease according to propositions 7.5 and 7.6, one need to add/update some entries in the structure `BoundPerPosExt`. For that, only two types of patterns should be looked for:

1. The new *cotp* patterns mined by REFINEANDMINE, that is those which left or right bound on attribute $attr$ is the added value cut. Visiting these patterns will add potentially new entries in `BoundPerPosExt` or update ancient ones.
2. The old *cotp* which core changes (i.e. becomes less restrictive) in the new discretization. One can show that these patterns are those which left bound is $prev(cut, C^{cur}_{attr})$ or right

bound is $next(cut, C^{cur}_{attr})$ on attribute $attr$. Visiting these patterns will only update ancient entries of `BoundPerPosExt` by potentially decreasing both second and third value.

7.3.3.4 Adding a new cut-point (line 7)

We have implemented a strategy which aims to decrease the $\overline{accuracy_\phi}$. For that, we search in `BoundPerPosExt` for the description d having the maximal value $\phi\big(fpr\big(core(d)\big), tpr\big(d\big)\big)$. In order to decrease $\overline{accuracy_\phi}$, we increase the size of $core(d)$ (to potentially increase $fpr\big(core(d)\big)$). This is equivalent to choose a cut-point in the border region $C^{rel}_{attr} \setminus C^{cur}_{attr}$ for some attribute $attr$ such that $cut \in d.I_{attr} \setminus core(d).I_{attr}$. Consider that we are in the step where the current discretization C^{cur} is the one depicted in Fig. 7.1. Imagine that the bottom pattern $\bot = \mathbb{R}^2$ is the one associated to the maximal value $\phi\big(tpr\big(\bot\big), fpr\big(core(\bot)\big)\big)$. The new cut-point should be chosen in $\{4,5\}$ for $attr = 1$ (recall that $core(\bot) = (-\infty, 3) \times (-\infty, +\infty)$). Note that if for such description there is no remaining relevant cut in its *border regions* for all $attr \in \{1, ..., |\mathcal{M}|\}$ then $core(d) = d$ ensuring that d is the *top pattern*.

7.4 Empirical Evaluation

In this section we report quantitative experiments over the implemented algorithms. For reproducibility purpose, the source code is made available in:

`https://github.com/Adnene93/RefineAndMine`

Experiments were carried out on a variety of datasets (Tab. 7.1) involving ordinal or continuous numerical attributes from the UCI repository. More extensive experiments are provided in the technical report `https://hal.archives-ouvertes.fr/hal-01874949/document`.

First, we study the effectiveness of REFINEANDMINE in terms of the speed of convergence to the optimal solution, as well as regarding the evolution over time of the accuracy of the provided bounding quality's guarantee. To this end, we report in Fig. 7.3, the behavior of REFINEANDMINE (i.e. quality and bounding guarantee) according to the execution time to evaluate the time/quality trade-off of the devised approach. $\overline{accuracy}$ as presented in Theorem 7.1 is the difference

Dataset	num	rows	intervals	class	α
HABERMAN_03_2	3	306	47×10^6	2	0.26
GLASS_04_1	4	214	5×10^{15}	1	0.33
ABALONE_02_M	2	4177	56×10^6	M	0.37
CREDITA_02_+	2	666	1×10^9	+	0.45

Table 7.1: Benchmark datasets and their characteristics: number of numerical attributes, number of rows, number of all possible intervals, the considered class and its prevalence (α)

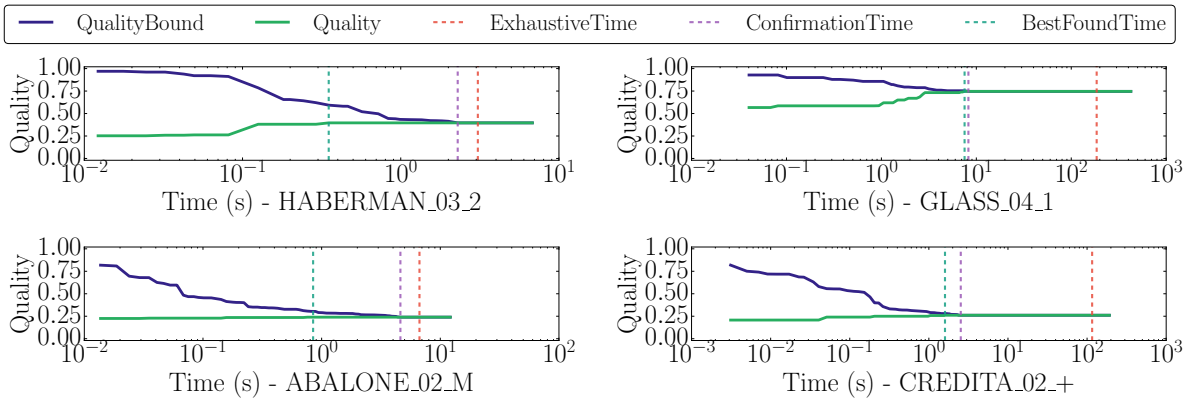


Figure 7.3: Evolution over time of top pattern quality and its bounding guarantee provided by REFINEANDMINE. Execution time is reported in log scale.

between the quality and its bounding measure. The experiments were conducted by running both REFINEANDMINE and the exhaustive enumeration algorithm (MININTCHANGE performed considering $\mathcal{D}_{C^{rel}}$) on the benchmark datasets using *informedness* measure. The exhaustive algorithm execution time enables the estimation of the computational overhead incurred by REFINEANDMINE. We interrupt a method if its execution time exceeds two hours. Note that, in the experiments, we choose to disable the computation of specificity since the latter is only optional and does not affect the effectiveness of the algorithm. This in contrast to the quality bound computation which is essential as it guides REFINEANDMINE in the cut-points selection strategy. The experiments give evidence of the effectiveness of REFINEANDMINE both in terms of finding the optimal solution as well as in providing stringent bound on the top quality pattern in a prompt manner. Two important milestones achieved by REFINEANDMINE during its execution are highlighted in Fig. 7.3. The first one, illustrated by the *green dotted line*, points out the required time to find the best pattern. The second milestone (*purple line*) is reached when the quality's and the bound's curves meet, this ensures that the best quality was already found by REFINEANDMINE. Interestingly, we observe that for most configurations the second milestone is attained by REFINEANDMINE promptly and well before the exhaustive method termination time. This is explained by the fact that the adopted cut points selection strategy aims to decrease as early as possible the $\overline{accuracy}$ metric. Finally, REFINEANDMINE requires in average 2 times of the requested execution time (*red dotted line*) by the exhaustive algorithm. This overhead is mostly incurred by the quality guarantee computation.

We illustrate in Fig. 7.4 the behavior of REFINEANDMINE in terms of finding diverse set of high quality patterns covering different parts of the dataset. To evaluate how quickly the devised approach finds a diverse patterns set, we run the exhaustive approach over the benchmark datasets to constitute a top-k diverse patterns set heuristically as following: the patterns extracted by the exhaustive search algorithm are sorted according to the quality measure and the best pattern is kept in the returned top-k list. Next, the complete patterns list are iterated over, and

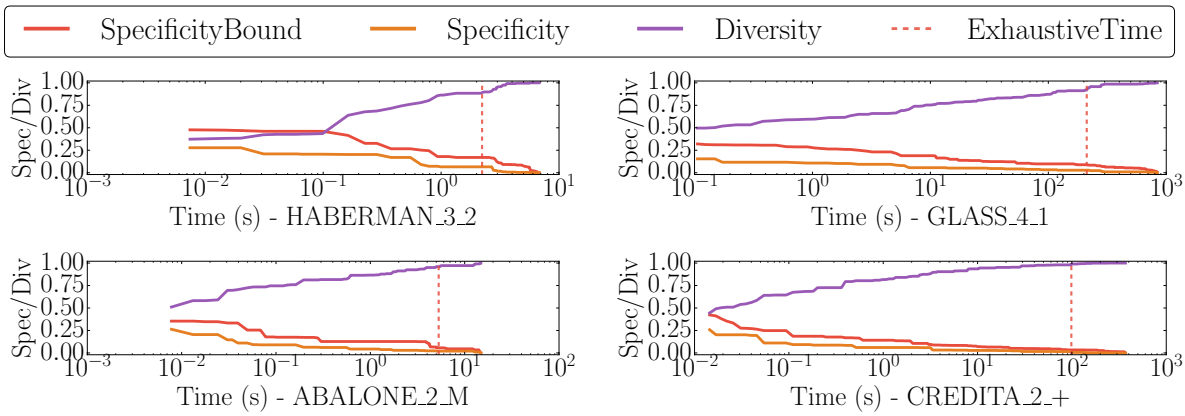


Figure 7.4: Efficiency of REFINEANDMINE in terms of retrieving a diverse patterns set. Execution time is reported in log scale. The ground-truth for each benchmark dataset corresponds to the obtained Top10 diversified patterns set with a similarity threshold of 0.25 and a minimum tpr of 15% .

the top-k list is augmented by a pattern if and only if its similarity with all the patterns of the current content of the top-k list is lower than a given threshold (a Jaccard index between extents). This process is interrupted if the desired number of patterns of the top-k list is reached or no remaining dissimilar pattern is available. Similar post-processing techniques were used by [34, 160]. Once this ground truth top-k list is constituted over some benchmark dataset, we run REFINEANDMINE and measure the *specificity quantity* of the obtained results set Sol with the top-k list. *specificity* metric is rewritten in eq. 7.1 to accommodate the desired evaluation objective of these experiments. Still, it remains upper-bounded by the general formula of $\overline{specificity}$ given in Theorem 7.2. This in order to evaluate at what extent the visited patterns by REFINEANDMINE well-cover the ground-truth patterns which are scattered over different parts of some input dataset. We report in Fig. 7.4 both *specificity* and its bounding guarantee $\overline{specificity}$, as well as, a diversity metric defined in eq. 7.2. Such a metric was defined in [34] to evaluate the ability of an approximate algorithm to retrieve a given ground-truth (i.e. diversified top-k discriminant patterns set). This diversity metric relies on a similarity rather than a distance (as in specificity), and is equal to 1 when all patterns of the top-k list are fully discovered.

$$specificity(\text{top-k}, Sol) = \sup_{d \in \text{top-k}} \inf_{c \in Sol} \left(|ext(d) \Delta ext(c)| / |\mathcal{G}| \right) \tag{7.1}$$

$$diversity(\text{top-k}, Sol) = \operatorname{avg}_{d \in \text{top-k}} \sup_{c \in Sol} \left(Jaccard(ext(d), ext(c)) \right) \tag{7.2}$$

Where the measure $Jaccard$ is given by:

$$Jaccard : \wp(\mathcal{G}) \times \wp(\mathcal{G}) \rightarrow [0,1], A, B \mapsto \frac{|A \cap B|}{|A \cup B|}$$

In most configurations, we notice that REFINEANDMINE is able to uncover approximately 80% (given by *diversity*) of the ground truth's patterns in less than 20% of the time required

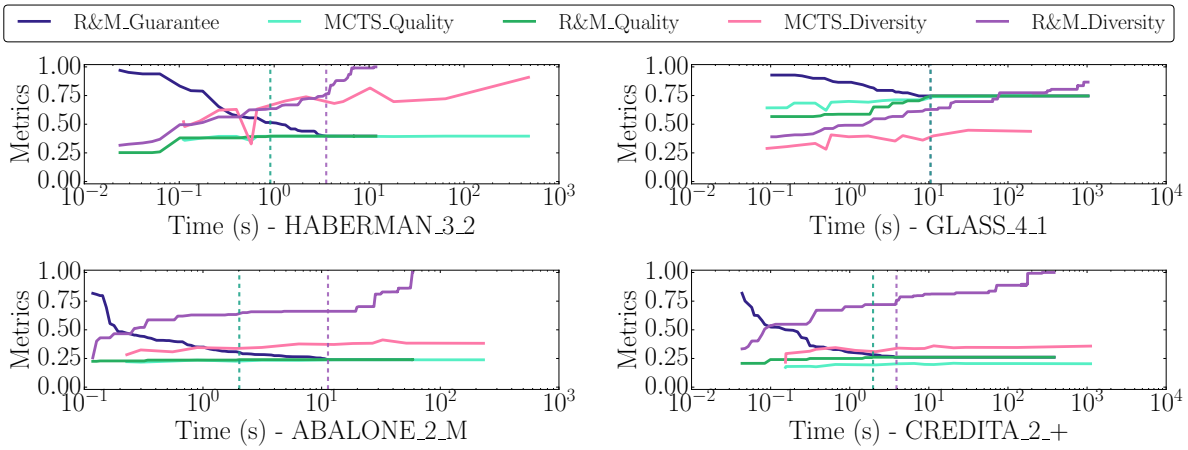


Figure 7.5: Comparative experiments between REFINEANDMINE ($R\&M$) and MCTS4DM. Execution time is reported in log scale. The ground-truth for each benchmark dataset corresponds to the obtained Top10 diversified patterns set with a similarity threshold of 0.25 and no minimum support size threshold.

by the exhaustive search algorithm. For instance, in ABALONE_02_M, we observe that after 2 seconds (12% of the required time for the exhaustive algorithm), the patterns outputted by REFINEANDMINE approximate 92% of the ground truth. Moreover, we observe that the specificity and $\overline{specificity}$ decrease quickly with time, guaranteeing a high level of diversity.

For a comparative study, we choose to compare REFINEANDMINE with the closest approach following the same paradigm (anytime) in the literature, that is the recent MCTS4DM technique [34]. MCTS4DM is depicted by the authors as an algorithm which enables the anytime discovery of a diverse patterns set of high quality. While MCTS4DM ensures interruptibility and an exhaustive exploration if given enough time and memory budget, it does not ensures any theoretical guarantees on the distance from optimality and on the diversity. We report in Fig. 7.5 a comparative evaluation between the two techniques. To realize this study, we investigate the ability of the two methods in retrieving the ground truth patterns, this by evaluating the quality of their respective diversified top-k lists against the ground truth using the diversity metric (eq. 7.2). We observe that REFINEANDMINE outperforms MCTS4DM both in terms of finding the best pattern, and of uncovering diverse patterns set of high qualities. This is partially due to the fact that our method is specifically tailored for mining discriminant patterns in numerical data, in contrast to MCTS4DM which is agnostic of the interestingness measure and the description language. Note that, to enable a fair comparison of the two approaches, we report the full time spent by the methods including the overhead induced by the post-computation of the diversified top-k patterns set.

7.5 Conclusion

We studied here in this chapter a novel anytime pattern mining technique for uncovering discriminant subgroups in numerical data. By leveraging the properties of the quality measures, we defined a guarantee on the accuracy of REFINEANDMINE in approximating the optimal solution which improves over time. We also presented a guarantee on the specificity of REFINEANDMINE –which is agnostic of the quality measure– ensuring its diversity and completeness. Empirical evaluation gives evidence of the effectiveness both in terms of finding the optimal solution (w.r.t. the quality measure ϕ) and revealing local optima located in different parts of the data.

This work paves the way for many improvements. REFINEANDMINE can be initialized with more sophisticated discretization techniques [63, 108]. Moreover, one can accelerate the algorithms by investigating other cut-points selection strategies.

While we considered here discriminative subgroup discovery, the enumeration process (i.e. *successive refinement of discretizations*) can be tailored to various other quality measures in subgroup discovery. For example, the accuracy bound guarantee definition can be easilly extended to handle several other traditional measures such as Mutual Information, χ^2 and $Gini\ split$ by exploiting their (quasi)-convexity properties w.r.t. tpr and fpr variables [1, 132]. Other improvements include the adaptation of REFINEANDMINE for high-dimensional datasets and its generalization for handling additional types of attributes (categorical, itemsets, etc.). The latter is facilitated by the generic notions from Section 7.3 and the recent works of Buzmakov et al. under the name of $\Sigma o \phi \iota \alpha$ [40, 42, 43].

Last but not least, a more interesting anytime algorithm for discriminative subgroup discovery would only output a small set of discriminative subgroups. Providing a form of guarantee on the small set of discriminative subgroups depends on the formalization of the problem and need to be thoroughly investigated.

CHAPTER 8

CONCLUSION AND FUTURE DEVELOPMENTS

8.1 Summary

8.1.1 Understanding Pattern Languages

The starting point of this thesis was to investigate other description languages than intervals when numerical attributes are considered. To this aim, we have studied and proposed the pattern language of conjunction of linear inequalities [20] which was formally more expressive. However, this pattern language showed some issues: (1) first, the number of subgroups induced by this pattern language is extremely huge and (2) second, the lack of interpretability of the pattern language. Indeed, a subgroup separable with a conjunction of 500 linear inequalities cannot be considered as intelligible. To address these problems, we have suggested that one should limit the number of linear inequalities to control the intelligibility of the pattern language while loosing some expressivity. However, once the number of linear inequalities was limited, we lost the lattice structure. Indeed, let us consider for instance the language $\mathcal{D}_h$ given by all singletons in $\wp(\mathbb{R}^2)$ and the half-planes in $\wp(\mathbb{R}^2)$, formally:

$$\mathcal{D}_h := \left\{\{z\} \mid z \in \mathbb{R}^2\right\} \cup \left\{\left\{(x,y) \in \mathbb{R}^2 \mid a \cdot x + b \cdot y \leq c\right\} \mid a,b,c \in \mathbb{R}\right\}$$

Poset $(\mathcal{D}_h, \supseteq)$ does not form a meet-semilattice. Same observation can be made on neighborhood pattern language [86] $\mathcal{D}_n$ given by the set of all possible closed disks in $\mathbb{R}^2$, formally:

$$\mathcal{D}_n := \left\{\left\{(x,y) \in \mathbb{R}^2 \mid (x-a)^2 + (y-b)^2 \leq c^2\right\} \mid a,b,c \in \mathbb{R}\right\}$$

If we consider that objects are described by points in $\mathbb{R}^2$, subgroups separable by $\mathcal{D}_h$ (resp. $\mathcal{D}_n$) are said to be linearly (resp. circularly) separable.

Since these languages are not meet-semilattices, they do not induce pattern structures and therefore one cannot use usual algorithms based on closure to enumerate exhaustively and non-redundantly the set of all possible subgroups separable by these languages. After investigating how non-lattice languages are considered in the formal concept analysis literature, we found that, often, they are transformed to pattern structures thanks to completions as it is the case for sequential patterns [39, 146] and graph patterns [78]. However, one can show that a finite set of points is separable by a convex set if and only if it is separable by an intersection of elements in $\mathcal{D}_h$ since convex polygons are intersection of half-planes. Same observation holds for the space $\mathcal{D}_n$ since (1) intersection of an arbitrary set of disks is convex and (2) if a finite set of points is linearly separable then it also circularly separable. In other words, completions of finite pattern setups induced by these languages build the convex set pattern structure making such a technique of transformation not an option.

We needed hence to study these languages from an order-theoretic point of view rather than systematically transform them to lattices. We drafted then what we called *pattern contexts* since then. Interestingly, SERGEI O. KUZNETSOV told us that pattern contexts was by definition *pattern setups* coined by LARS LUMPE and STEFAN SCHMIDT [123] but sadly not studied. In response, we started a preliminary study on this formalism under the name of *pattern setups and their completions* [18]. One of the important result in this latter paper is that using the antichain embedding [39, 78] to transform pattern setups to pattern structures was not always applicable unless we deal with what we called *hyper-lattices*[1]. Interestingly, after discussing our work with the CLA community, STEFAN SCHMIDT attracted our attention to the work of MIHAIL BENADO on multilattices [22] which, indeed, were tightly linked with what we called *hyper-lattices*. Connection to multilattices and multisemilattices [128] were made after. Later discussions with JOZEF PÓCS helped us to establish also connections between chain-complete posets and complete multilattices.

8.1.2 Tackling Subgroup enumeration in Interval Pattern Structures

Beside understanding pattern languages, we were interested by the exhaustive enumeration of subgroups. Particularly, for the interval pattern language, KAYTOUE ET AL. [104] showed that algorithm MININTCHANGE outperformed even the fastest known implementation of CLOSE-BY-ONE (CBO), namely LCMV2 [157], when enumerating closed interval patterns in interordinal scaled contexts [79]. When analyzing what make MININTCHANGE faster, we found that MININTCHANGE avoids generating some closed itemsets comparing to CBO by reordering them. This reordering is directly linked to leveraging inherent implications between items in the interordinal scaled context. We proposed then Algorithm CLOSE-BY-ONE USING IMPLICATIONS (CBOI) [16] that enumerate closed itemsets (and equivalently subgroups) in contexts using the inherent implications existing between attributes. CBOI behaves in the interordinal scaled context as

[1]The term *"Hyper-lattice"* was introduced in [168] and is linked to the antichain completion of a poset.

MININTCHANGE does in the interval pattern structure. Moreover, CBOI can be seen as a generalization of other algorithms like CELLIER ET AL.'s [46] algorithm for mining closed itemsets when an attribute taxonomy is provided.

8.1.3 Investigating Discriminative Subgroup Discovery

The main target application that we have considered when formalizing pattern languages was the task of *discriminative subgroup discovery*.

We have been tempted in the beginning to investigate the problem in an *exact way*, i.e. find the best subgroup optimizing some quality measure, *a priori* a probability-based measure having the weak dominance property (cf. Definition 6.9). Connections to Garriga's relevant patterns [82] became evident[2]. However, even if using relevance theory allows to reduce significantly the number of subgroups, the problem remained intractable. Henceforth, the necessity of using approximate algorithm to handle bigger datasets revealed to be unavoidable.

Several heuristic algorithms have been proposed in the literature to solve the problem of discriminative subgroup discovery. We cite as a matter of example Beam Search Techniques [59, 115, 159, 160], Evolutionary Algorithms [44], Sampling techniques [23, 30, 31] and Anytime algorithms [34]. However, while these algorithms showed their effectiveness and efficiency to solve this problem, the user cannot have any idea of how the already best found subgroup approximates the best quality subgroup hidden in the dataset. We argued then, perhaps naively[3], that the best trade-off between exact algorithms and non-exact ones are anytime algorithms providing guarantees on the output subgroups upon interruption. A first attempt of such an anytime algorithm, namely REFINEANDMINE, have been proposed and presented in ECML/PKDD'18 [15]. The algorithm was tailored for interval patterns in dataset with exclusively numerical attributes. It does provide two guarantees on the output subgroups upon interruption. The first evaluates how close is the best found subgroup so far to the optimal one in the whole search space. The second measures how already found subgroups are diverse and cover well all the interesting regions in the dataset.

8.2 Perspectives and Open Problems

The different investigations made in this work suggest diverse questions that, to the best of our knowledge, remain open.

8.2.1 On Pattern Setups Transformations

Projections are a well-founded tool to transform pattern setups to simpler ones. However, we find such a tool rather limiting. For instance, if we consider the pattern setup induced by neighborhood

[2] It was THOMAS GUYET presentation's at ICFCA'2017 [92] that drew our attention to Garriga's relevant patterns.
[3] Is it intractable to find a good approximate solution?

patterns and the convex set pattern structure, it is clear that any subgroup separable in the former pattern setup is also separable in the latter one. Since projections maintain pattern structures, convex set pattern structures can not be projected to the pattern setup of neighborhood patterns. Moreover, recall that interval pattern structure can be seen as a projection of the convex set pattern structure (see Section 5.5.3). To do that, it was necessary that the language of interval patterns form a suborder on the language of convex set patterns which is rather limiting. Indeed, if we used a different formalism to model the interval pattern structure (see Note 5.6), the interval pattern structure can longer be seen as a projection of the convex set pattern structure.

Following our previous observations, we argue that a more permissive formal tool is needed to *simplify* pattern setups. Interestingly, BUZMAKOV [38] formulated somehow the same perspective. LUMPE AND SCHMIDT [123] provide an initial unifying view on pattern setup transformations.

8.2.2 On Enumerating Subgroups

As we have discussed earlier in the conclusion of Chapter 5, elaborating better techniques to enumerate subgroups exhaustively and non-redundantly for a given pattern setup is needed. Such techniques will be useful to enumerate subgroups induced by pattern languages like *sequential patterns* [6], *graph patterns* [110] or *neighborhood patterns* [86].

The particular instance of sequence over itemsets. One should note that, even if we consider the well-investigated language of *"sequence over itemsets"* pattern language [6], the existence of a POLY-DELAY algorithm enumerating all subgroups induced by such a language remains open.

Another question that we find important is the following one: *"Let G be a finite set of objects and let S be a set-system on G s.t. the right-hand side of Proposition 3.5 holds. Is there a pattern setup $\mathbb{P}$ over the description language of sequences over itemsets s.t. $S = \mathbb{P}_{ext}$?"*. A negative answer to this question allows us to understand if the space of subgroups induced by sequences over itemsets language has some structure or not. Such a structure can be potentially leveraged for the enumeration of subgroups induced by this language.

8.2.3 On Enumerating Relevant Subgroups in Labeled Datasets

One important problem arising when dealing with labeled datasets is the problem of enumerating relevant subgroups exhaustively and non-redundantly as they represent, objectively, the minimal set of subgroups to be looked for when labeled datasets are considered. The best algorithms proposed in the literature was proposed by HENRIK GROSSKREUTZ [88] when itemset pattern language is considered. He proposed particularly two enumeration techniques. The first one is POLY-OUTPUT but not PSPACE while the second one is PSPACE but not POLY-OUTPUT. The existence of a POLY-OUTPUT and PSPACE algorithm, or better a POLY-DELAY one, for relevant subgroup enumeration remains an open problem.

8.2.4 On Finding the Best Quality Subgroups in Labeled Datasets

Given a pattern language and a quality measure, particularly a probability-based measure having the weak dominance property (cf. Definition 6.9), what is the complexity of the problem of finding the best-quality subgroup? Surprisingly, even with the myriad of algorithms that have been proposed in the literature to tackle this problem, its complexity remains open. Showing the tractability or the intractability of the problem brings a solid justification for why elaborating better non-exact algorithms is crucial.

Interestingly, ANGLUIN AND LIARD [9] showed that, particularly for the *accuracy* measure (see Table 6.2) and the language of itemsets, this problem is NP-Complete (see Theorem 4 in [9]). BEN-DAVID [21] showed that this same problem is even hard to approximate when a certain minimal ratio is desired for several pattern languages including itemset and interval pattern languages. These works give a good starting point to tackle other quality measures in discriminative subgroup discovery literature as for instance the *weighted relative accuracy (WRAcc)* measure [114].

8.2.5 On Elaborating Anytime Subgroup Discovery Techniques

We argued that the best trade-off between exact algorithms and non-exact algorithms for subgroup discovery in labeled datasets are anytime algorithms that provide guarantees upon interruption. We tackled the only problem of finding subgroups induced by the interval pattern language in [15]. Extending this algorithm to handle other pattern languages like the itemset one is one of our main preoccupation currently.

Moreover, Subgroup Discovery is not only about finding best-quality subgroups. It is also about finding several subgroups covering different regions in the dataset, i.e. a diverse subgroup set. Designing anytime algorithms that provide form of guarantees on an output diverse subgroup set is also an important problem that should be thoroughly investigated.

BIBLIOGRAPHY

[1] T. ABUDAWOOD AND P. A. FLACH, *Evaluation measures for multi-class subgroup discovery*, in ECML PKDD, Springer, 2009, pp. 35–50.

[2] M. ADDA, L. WU, AND Y. FENG, *Rare itemset mining*, in ICMLA, IEEE Computer Society, 2007, pp. 73–80.

[3] P. AGARWAL, M. KAYTOUE, S. O. KUZNETSOV, A. NAPOLI, AND G. POLAILLON, *Symbolic galois lattices with pattern structures*, in RSFDGrC, vol. 6743 of Lecture Notes in Computer Science, Springer, 2011, pp. 191–198.

[4] R. AGRAWAL, T. IMIELINSKI, AND A. N. SWAMI, *Mining association rules between sets of items in large databases*, in SIGMOD Conference, ACM Press, 1993, pp. 207–216.

[5] R. AGRAWAL AND R. SRIKANT, *Mining sequential patterns*, in ICDE, IEEE Computer Society, 1995, pp. 3–14.

[6] R. AGRAWAL, R. SRIKANT, ET AL., *Mining sequential patterns*, in icde, vol. 95, 1995, pp. 3–14.

[7] A. V. AHO, M. R. GAREY, AND J. D. ULLMAN, *The transitive reduction of a directed graph*, SIAM J. Comput., 1 (1972), pp. 131–137.

[8] S. ANDREWS, *A new method for inheriting canonicity test failures in close-by-one type algorithms*, in CLA, vol. 2123 of CEUR Workshop Proceedings, CEUR-WS.org, 2018, pp. 255–266.

[9] D. ANGLUIN AND P. D. LAIRD, *Learning from noisy examples*, Machine Learning, 2 (1987), pp. 343–370.

[10] M. ATZMÜLLER AND F. PUPPE, *Sd-map - A fast algorithm for exhaustive subgroup discovery*, in PKDD, vol. 4213 of Lecture Notes in Computer Science, Springer, 2006, pp. 6–17.

[11] J. BAIXERIES, M. KAYTOUE, AND A. NAPOLI, *Computing functional dependencies with pattern structures*, in CLA, vol. 972, 2012, pp. 175–186.

[12] ——, *Characterizing functional dependencies in formal concept analysis with pattern structures*, Ann. Math. Artif. Intell., 72 (2014), pp. 129–149.

[13] S. D. BAY AND M. J. PAZZANI, *Detecting group differences: Mining contrast sets*, Data Min. Knowl. Discov., 5 (2001), pp. 213–246.

[14] R. J. BAYARDO AND R. AGRAWAL, *Mining the most interesting rules*, in KDD, ACM, 1999, pp. 145–154.

[15] A. BELFODIL, A. BELFODIL, AND M. KAYTOUE, *Anytime subgroup discovery in numerical domains with guarantees*, in ECML/PKDD (2), vol. 11052 of Lecture Notes in Computer Science, Springer, 2018, pp. 500–516.

[16] ——, *Mining formal concepts using implications between items*, in ICFCA, vol. 11511 of Lecture Notes in Computer Science, Springer, 2019, pp. 173–190.

[17] A. BELFODIL, S. CAZALENS, P. LAMARRE, AND M. PLANTEVIT, *Flash points: Discovering exceptional pairwise behaviors in vote or rating data*, in ECML/PKDD (2), 2017, pp. 442–458.

[18] A. BELFODIL, S. O. KUZNETSOV, AND M. KAYTOUE, *Pattern setups and their completions*, in CLA, vol. 2123 of CEUR Workshop Proceedings, CEUR-WS.org, 2018, pp. 243–253.

[19] ——, *On pattern setups and pattern multistructures*, CoRR, abs/1906.02963 (2019).

[20] A. BELFODIL, S. O. KUZNETSOV, C. ROBARDET, AND M. KAYTOUE, *Mining convex polygon patterns with formal concept analysis*, in IJCAI, ijcai.org, 2017, pp. 1425–1432.

[21] S. BEN-DAVID, N. EIRON, AND P. M. LONG, *On the difficulty of approximately maximizing agreements*, J. Comput. Syst. Sci., 66 (2003), pp. 496–514.

[22] M. BENADO, *Les ensembles partiellement ordonnés et le théorème de raffinement de schreier. ii. théorie des multistructures*, Czechoslovak Mathematical Journal, 5 (1955), pp. 308–344.

[23] A. A. BENDIMERAD, M. PLANTEVIT, AND C. ROBARDET, *Mining exceptional closed patterns in attributed graphs*, Knowl. Inf. Syst., 56 (2018), pp. 1–25.

[24] C. BERGE, *Hypergraphs: combinatorics of finite sets*, vol. 45, Elsevier, 1984.

[25] G. BIRKHOFF ET AL., *Rings of sets*, Duke Mathematical Journal, 3 (1937), pp. 443–454.

[26] A. BOGOMOLNY, *Cut The Knot: Perimeters of Convex Polygons, One within the Other (http://www.cut-the-knot.org/m/Geometry/ PerimetersOfTwoConvexPolygons.shtml)*, 2017.

[27] P. BOLDI AND S. VIGNA, *On the lattice of antichains of finite intervals*, CoRR, abs/1510.03675 (2016).

[28] M. BOLEY, T. HORVÁTH, A. POIGNÉ, AND S. WROBEL, *Efficient closed pattern mining in strongly accessible set systems (extended abstract)*, in PKDD, vol. 4702 of Lecture Notes in Computer Science, Springer, 2007, pp. 382–389.

[29] ——, *Listing closed sets of strongly accessible set systems with applications to data mining*, Theor. Comput. Sci., 411 (2010), pp. 691–700.

[30] M. BOLEY, C. LUCCHESE, D. PAURAT, AND T. GÄRTNER, *Direct local pattern sampling by efficient two-step random procedures*, in KDD, ACM, 2011, pp. 582–590.

[31] M. BOLEY, S. MOENS, AND T. GÄRTNER, *Linear space direct pattern sampling using coupling from the past*, in KDD, ACM, 2012, pp. 69–77.

[32] F. BONCHI AND C. LUCCHESE, *Extending the state-of-the-art of constraint-based pattern discovery*, Data Knowl. Eng., 60 (2007), pp. 377–399.

[33] J.-P. BORDAT, *Calcul pratique du treillis de galois d'une correspondance*, Mathématiques et Sciences humaines, 96 (1986), pp. 31–47.

[34] G. BOSC, J. BOULICAUT, C. RAÏSSI, AND M. KAYTOUE, *Anytime discovery of a diverse set of patterns with monte carlo tree search*, Data Min. Knowl. Discov., 32 (2018), pp. 604–650.

[35] S. BOYD AND L. VANDENBERGHE, *Convex optimization*, Cambridge university press, 2004.

[36] P. BRITO, *Order structure of symbolic assertion objects*, IEEE Trans. Knowl. Data Eng., 6 (1994), pp. 830–834.

[37] ——, *Symbolic data analysis: another look at the interaction of data mining and statistics*, Wiley Interdiscip. Rev. Data Min. Knowl. Discov., 4 (2014), pp. 281–295.

[38] A. BUZMAKOV, *Formal Concept Analysis and Pattern Structures for mining Structured Data. (Analyse formelle de concepts et structures de patrons pour la fouille de données structurées)*, PhD thesis, University of Lorraine, Nancy, France, 2015.

[39] A. BUZMAKOV, E. EGHO, N. JAY, S. O. KUZNETSOV, A. NAPOLI, AND C. RAÏSSI, *On mining complex sequential data by means of FCA and pattern structures*, Int. J. General Systems, 45 (2016), pp. 135–159.

[40] A. BUZMAKOV, S. O. KUZNETSOV, AND A. NAPOLI, *Fast generation of best interval patterns for nonmonotonic constraints*, in ECML/PKDD (2), vol. 9285 of Lecture Notes in Computer Science, Springer, 2015, pp. 157–172.

[41] ——, *Revisiting pattern structure projections*, in ICFCA, vol. 9113 of Lecture Notes in Computer Science, Springer, 2015, pp. 200–215.

[42] ——, *Efficient mining of subsample-stable graph patterns*, in ICDM, IEEE Computer Society, 2017, pp. 757–762.

[43] ——, *Mining best closed itemsets for projection-antimonotonic constraints in polynomial time*, CoRR, abs/1703.09513 (2017).

[44] C. J. CARMONA, P. GONZÁLEZ, M. J. DEL JESÚS, AND F. HERRERA, *Overview on evolutionary subgroup discovery: analysis of the suitability and potential of the search performed by evolutionary algorithms*, Wiley Interdiscip. Rev. Data Min. Knowl. Discov., 4 (2014), pp. 87–103.

[45] G. CASAS-GARRIGA, *Summarizing sequential data with closed partial orders*, in SDM, SIAM, 2005, pp. 380–391.

[46] P. CELLIER, S. FERRÉ, O. RIDOUX, AND M. DUCASSÉ, *An algorithm to find frequent concepts of a formal context with taxonomy*, in CLA, 2006, pp. 226–231.

[47] L. CERF, *Constraint-Based Mining of Closed Patterns in Noisy n-ary Relations. (Fouille Sous Contraintes de Motifs Fermés dans des Relations n-aires Bruitées)*, PhD thesis, INSA de Lyon, Lyon - Villeurbanne, France, 2010.

[48] V. CODOCEDO, G. BOSC, M. KAYTOUE, J. BOULICAUT, AND A. NAPOLI, *A proposition for sequence mining using pattern structures*, in ICFCA, 2017.

[49] V. CODOCEDO AND A. NAPOLI, *Lattice-based biclustering using partition pattern structures*, in ECAI, vol. 263 of Frontiers in Artificial Intelligence and Applications, IOS Press, 2014, pp. 213–218.

[50] P. CORDERO, G. GUTIÉRREZ, J. MARTÍNEZ, AND I. P. DE GUZMÁN, *A new algebraic tool for automatic theorem provers*, Annals of Mathematics and Artificial Intelligence, 42 (2004), pp. 369–398.

[51] J. CRAMPTON AND G. LOIZOU, *The completion of a poset in a lattice of antichains*, International Mathematical Journal, 1 (2001), pp. 223–238.

[52] B. CRÉMILLEUX, A. GIACOMETTI, AND A. SOULET, *How your supporters and opponents define your interestingness*, in ECML/PKDD (1), vol. 11051 of Lecture Notes in Computer Science, Springer, 2018, pp. 373–389.

[53] B. A. DAVEY AND H. A. PRIESTLEY, *Introduction to lattices and order*, Cambridge university press, 2002.

[54] K. DENECKE AND S. L. WISMATH, *Galois connections and complete sublattices*, in Galois Connections and Applications, Springer, 2004, pp. 211–229.

[55] O. DEVILLERS, *On Deletion in Delaunay Triangulations*, in Proceedings of the Fifteenth Annual Symposium on Computational Geometry, Miami Beach, Florida, USA, June 13-16, 1999, V. Milenkovic, ed., ACM, 1999, pp. 181–188.

[56] J.-P. DOIGNON AND J.-C. FALMAGNE, *Knowledge spaces and learning spaces*, arXiv preprint arXiv:1511.06757, (2015).

[57] G. DONG AND J. LI, *Efficient mining of emerging patterns: Discovering trends and differences*, in KDD, ACM, 1999, pp. 43–52.

[58] W. DUIVESTEIJN, A. FEELDERS, AND A. J. KNOBBE, *Exceptional model mining - supervised descriptive local pattern mining with complex target concepts*, Data Min. Knowl. Discov., 30 (2016), pp. 47–98.

[59] W. DUIVESTEIJN, A. J. KNOBBE, A. FEELDERS, AND M. VAN LEEUWEN, *Subgroup discovery meets bayesian networks – an exceptional model mining approach*, in ICDM, IEEE Computer Society, 2010, pp. 158–167.

[60] P. H. EDELMAN AND R. E. JAMISON, *The theory of convex geometries*, Geometriae dedicata, 19 (1985), pp. 247–270.

[61] C. EVERETT, *Closure operators and galois theory in lattices*, Transactions of the American Mathematical Society, 55 (1944), pp. 514–525.

[62] W. B. EWALD, *From kant to hilbert, volume 2: A source book in the foundations of mathematics*, (2005).

[63] U. M. FAYYAD AND K. B. IRANI, *Multi-interval discretization of continuous-valued attributes for classification learning*, in IJCAI., 1993, pp. 1022–1029.

[64] U. M. FAYYAD, G. PIATETSKY-SHAPIRO, AND P. SMYTH, *From data mining to knowledge discovery in databases*, AI Magazine, 17 (1996), pp. 37–54.

[65] D. R. FENO, *Mesures de qualité des règles d'association : normalisation et caractérisation des bases*, PhD thesis, University of La Réunion, Saint-Denis, France, 2007.

[66] S. FERRÉ AND O. RIDOUX, *A logical generalization of formal concept analysis*, in ICCS, vol. 1867 of Lecture Notes in Computer Science, Springer, 2000, pp. 371–384.

[67] V. FINN, *On machine-oriented formalization of plausible reasoning in the style of f*, Bacon and DS Mill [in Russian], Semiotika i Informatika, 20 (1983), pp. 35–101.

[68] ——, *Plausible reasoning in systems of jsm type*, Itogi Nauki i Tekhniki, Seriya Informatika, 15 (1991), pp. 54–101.

[69] C. FLAMENT, *L'analyse booléenne de questionnaires*, Mathématiques et Sciences humaines, 12 (1965), pp. 3–10.

[70] P. FOURNIER-VIGER, J. C.-W. LIN, R. NKAMBOU, B. VO, AND V. S. TSENG, *High-Utility Pattern Mining*, Springer, 2019.

[71] W. J. FRAWLEY, G. PIATETSKY-SHAPIRO, AND C. J. MATHEUS, *Knowledge discovery in databases: An overview*, AI Magazine, 13 (1992), pp. 57–70.

[72] K. FUKUDA ET AL., *Frequently asked questions in polyhedral computation*, Swiss Federal Institute of Technology, Lausanne and Zurich, Switzerland. Available at: ftp://ftp. ifor. math. ethz. ch/pub/fukuda/reports/polyfaq040618. pdf, (2004).

[73] J. FÜRNKRANZ AND P. A. FLACH, *ROC 'n' rule learning - towards a better understanding of covering algorithms*, Machine Learning, 58 (2005), pp. 39–77.

[74] E. GALBRUN, P. CELLIER, N. TATTI, A. TERMIER, AND B. CRÉMILLEUX, *Mining periodic patterns with a MDL criterion*, in ECML/PKDD (2), vol. 11052 of Lecture Notes in Computer Science, Springer, 2018, pp. 535–551.

[75] E. GALBRUN AND P. MIETTINEN, *Redescription mining: An overview*, IEEE Intelligent Informatics Bulletin, 18 (2017), pp. 7–12.

[76] B. GANTER, *Two basic algorithms in concept analysis*, Technical report, Technische Hoschule Darmstadt, (1984).

[77] ——, *Order-embedded complete lattices*, in CLA, vol. 2123 of CEUR Workshop Proceedings, CEUR-WS.org, 2018, pp. 153–165.

[78] B. GANTER AND S. O. KUZNETSOV, *Pattern structures and their projections*, in ICCS, vol. 2120 of Lecture Notes in Computer Science, Springer, 2001, pp. 129–142.

[79] B. GANTER AND R. WILLE, *Conceptual scaling*, in Applications of combinatorics and graph theory to the biological and social sciences, 1989, pp. 139–167.

[80] B. GANTER AND R. WILLE, *Formal concept analysis - mathematical foundations*, Springer, 1999.

[81] G. C. GARRIGA, P. KRALJ, AND N. LAVRAC, *Closed sets for labeled data*, in PKDD, vol. 4213 of Lecture Notes in Computer Science, Springer, 2006, pp. 163–174.

[82] ——, *Closed sets for labeled data*, Journal of Machine Learning Research, 9 (2008), pp. 559–580.

[83] A. GÉLY, *A generic algorithm for generating closed sets of a binary relation*, in ICFCA, vol. 3403 of Lecture Notes in Computer Science, Springer, 2005, pp. 223–234.

[84] L. GENG AND H. J. HAMILTON, *Interestingness measures for data mining: A survey*, ACM Comput. Surv., 38 (2006), p. 9.

[85] A. GIACOMETTI, D. H. LI, P. MARCEL, AND A. SOULET, *20 years of pattern mining: a bibliometric survey*, SIGKDD Explorations, 15 (2013), pp. 41–50.

[86] A. GIACOMETTI AND A. SOULET, *Dense neighborhood pattern sampling in numerical data*, in SDM, SIAM, 2018, pp. 756–764.

[87] F. GIANNOTTI, M. NANNI, F. PINELLI, AND D. PEDRESCHI, *Trajectory pattern mining*, in KDD, ACM, 2007, pp. 330–339.

[88] H. GROSSKREUTZ, *Class relevant pattern mining in output-polynomial time*, in SDM, SIAM / Omnipress, 2012, pp. 284–294.

[89] H. GROSSKREUTZ AND S. RÜPING, *On subgroup discovery in numerical domains*, Data Min. Knowl. Discov., 19 (2009), pp. 210–226.

[90] H. GROSSKREUTZ, S. RÜPING, AND S. WROBEL, *Tight optimistic estimates for fast subgroup discovery*, in ECML/PKDD (1), vol. 5211 of Lecture Notes in Computer Science, Springer, 2008, pp. 440–456.

[91] J.-L. GUIGUES AND V. DUQUENNE, *Familles minimales d'implications informatives résultant d'un tableau de données binaires*, Mathématiques et Sciences humaines, 95 (1986), pp. 5–18.

[92] T. GUYET, R. QUINIOU, AND V. MASSON, *Mining relevant interval rules*, CoRR, abs/1709.03267 (2017).

[93] M. R. HACENE, M. HUCHARD, A. NAPOLI, AND P. VALTCHEV, *Relational concept analysis: mining concept lattices from multi-relational data*, Ann. Math. Artif. Intell., 67 (2013), pp. 81–108.

[94] J. HAN, H. CHENG, D. XIN, AND X. YAN, *Frequent pattern mining: current status and future directions*, Data mining and knowledge discovery, 15 (2007), pp. 55–86.

[95] C. HÉBERT AND B. CRÉMILLEUX, *A unified view of objective interestingness measures*, in MLDM, vol. 4571 of Lecture Notes in Computer Science, Springer, 2007, pp. 533–547.

[96] F. HERRERA, C. J. CARMONA, P. GONZÁLEZ, AND M. J. DEL JESÚS, *An overview on subgroup discovery: foundations and applications*, Knowl. Inf. Syst., 29 (2011), pp. 495–525.

[97] J. HILLS, L. M. DAVIS, AND A. J. BAGNALL, *Interestingness measures for fixed consequent rules*, in IDEAL, vol. 7435 of Lecture Notes in Computer Science, Springer, 2012, pp. 68–75.

[98] Q. HU AND T. IMIELINSKI, *ALPINE: progressive itemset mining with definite guarantees*, in SDM, SIAM, 2017, pp. 63–71.

[99] D. P. HUTTENLOCHER, G. A. KLANDERMAN, AND W. RUCKLIDGE, *Comparing images using the hausdorff distance*, IEEE Trans. Pattern Anal. Mach. Intell., 15 (1993), pp. 850–863.

[100] T. IMIELINSKI AND H. MANNILA, *A database perspective on knowledge discovery*, Commun. ACM, 39 (1996), pp. 58–64.

[101] F. JANSSEN AND J. FÜRNKRANZ, *On trading off consistency and coverage in inductive rule learning*, in LWA, vol. 1/2006 of Hildesheimer Informatik-Berichte, University of Hildesheim, Institute of Computer Science, 2006, pp. 306–313.

[102] D. S. JOHNSON, C. H. PAPADIMITRIOU, AND M. YANNAKAKIS, *On generating all maximal independent sets*, Inf. Process. Lett., 27 (1988), pp. 119–123.

[103] B. KAVSEK, N. LAVRAC, AND V. JOVANOSKI, *APRIORI-SD: adapting association rule learning to subgroup discovery*, in IDA, vol. 2810 of Lecture Notes in Computer Science, Springer, 2003, pp. 230–241.

[104] M. KAYTOUE, S. O. KUZNETSOV, AND A. NAPOLI, *Revisiting numerical pattern mining with formal concept analysis*, in IJCAI, IJCAI/AAAI, 2011, pp. 1342–1347.

[105] M. KAYTOUE, M. PLANTEVIT, A. ZIMMERMANN, A. A. BENDIMERAD, AND C. ROBARDET, *Exceptional contextual subgraph mining*, Machine Learning, 106 (2017), pp. 1171–1211.

[106] W. KLÖSGEN, *Explora: A multipattern and multistrategy discovery assistant*, in Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press, 1996, pp. 249–271.

[107] P. KRAJCA, J. OUTRATA, AND V. VYCHODIL, *Advances in algorithms based on cbo*, in CLA, 2010, pp. 325–337.

[108] L. KURGAN AND K. J. CIOS, *Discretization algorithm that uses class-attribute interdependence maximization*, in IC-AI, 2001, pp. 980–987.

[109] S. O. KUZNETSOV, *A Fast Algorithm for Computing All Intersections of Objects in a Finite Semi-lattice*, Nauchno-Tekhnicheskaya Informatsiya, ser. 2 (1993), pp. 17–20.

[110] ——, *Learning of simple conceptual graphs from positive and negative examples*, in PKDD, 1999, pp. 384–391.

[111] ——, *Machine learning and formal concept analysis*, in ICFCA, vol. 2961 of Lecture Notes in Computer Science, Springer, 2004, pp. 287–312.

[112] ——, *Pattern structures for analyzing complex data*, in RSFDGrC, 2009.

[113] S. O. KUZNETSOV AND S. A. OBIEDKOV, *Comparing performance of algorithms for generating concept lattices*, J. Exp. Theor. Artif. Intell., 14 (2002), pp. 189–216.

[114] N. LAVRAC, P. A. FLACH, AND B. ZUPAN, *Rule evaluation measures: A unifying view*, in Inductive Logic Programming, 9th International Workshop, ILP-99, Bled, Slovenia, June 24-27, 1999, Proceedings, S. Dzeroski and P. A. Flach, eds., vol. 1634 of Lecture Notes in Computer Science, Springer, 1999, pp. 174–185.

[115] N. LAVRAC, B. KAVSEK, P. A. FLACH, AND L. TODOROVSKI, *Subgroup discovery with CN2-SD*, Journal of Machine Learning Research, 5 (2004), pp. 153–188.

[116] F. LE GALL, *Powers of tensors and fast matrix multiplication*, in Proceedings of the 39th international symposium on symbolic and algebraic computation, ACM, 2014, pp. 296–303.

[117] D. LEMAN, A. FEELDERS, AND A. J. KNOBBE, *Exceptional model mining*, in ECML/PKDD (2), vol. 5212 of Lecture Notes in Computer Science, Springer, 2008, pp. 1–16.

[118] F. LEMMERICH, M. ATZMUELLER, AND F. PUPPE, *Fast exhaustive subgroup discovery with numerical target concepts*, Data Min. Knowl. Discov., 30 (2016), pp. 711–762.

[119] F. LEMMERICH, M. ROHLFS, AND M. ATZMÜLLER, *Fast discovery of relevant subgroup patterns*, in FLAIRS Conference, AAAI Press, 2010.

[120] P. LENCA, P. MEYER, B. VAILLANT, AND S. LALLICH, *On selecting interestingness measures for association rules: User oriented description and multiple criteria decision aid*, European Journal of Operational Research, 184 (2008), pp. 610–626.

[121] B. LIU, W. HSU, AND Y. MA, *Integrating classification and association rule mining*, in KDD, AAAI Press, 1998, pp. 80–86.

[122] T. LUCAS, T. C. P. B. SILVA, R. VIMIEIRO, AND T. B. LUDERMIR, *A new evolutionary algorithm for mining top-k discriminative patterns in high dimensional data*, Appl. Soft Comput., 59 (2017), pp. 487–499.

[123] L. LUMPE AND S. E. SCHMIDT, *Pattern structures and their morphisms*, in CLA, vol. 1466 of CEUR Workshop Proceedings, CEUR-WS.org, 2015, pp. 171–179.

[124] M. LUXENBURGER, *Implications partielles dans un contexte*, Mathématiques et Sciences Humaines, 113 (1991), pp. 35–55.

[125] M. MAMPAEY, S. NIJSSEN, A. FEELDERS, AND A. J. KNOBBE, *Efficient algorithms for finding richer subgroup descriptions in numeric and nominal data*, in ICDM, 2012, pp. 499–508.

[126] M. MAMPAEY, S. NIJSSEN, A. FEELDERS, R. M. KONIJN, AND A. J. KNOBBE, *Efficient algorithms for finding optimal binary features in numeric and nominal labeled data*, Knowl. Inf. Syst., 42 (2015), pp. 465–492.

[127] H. MANNILA AND H. TOIVONEN, *Levelwise search and borders of theories in knowledge discovery*, Data Min. Knowl. Discov., 1 (1997), pp. 241–258.

[128] J. MARTÍNEZ, G. GUTIÉRREZ, I. DE GUZMÄN, AND P. CORDERO, *Multilattices via multisemilattices*, Topics in applied and theoretical mathematics and computer science, (2001), pp. 238–248.

[129] J. MARTÍNEZ, G. GUTIÉRREZ, I. P. DE GUZMÁN, AND P. CORDERO, *Generalizations of lattices via non-deterministic operators*, Discrete Mathematics, 295 (2005), pp. 107–141.

[130] A. MARY, *Énumération des dominants minimaux d'un graphe*, PhD thesis, Université Blaise Pascale - Clermont-Ferrand, 2013.

[131] J. W. MOON AND L. MOSER, *On cliques in graphs*, Israel journal of Mathematics, 3 (1965), pp. 23–28.

[132] S. MORISHITA AND J. SESE, *Traversing itemset lattice with statistical metric pruning*, in ACM SIGMOD-SIGACT-SIGART, 2000, pp. 226–236.

[133] S. MUGGLETON, *Inductive logic programming*, New Generation Comput., 8 (1991), pp. 295–318.

[134] P. K. NOVAK, N. LAVRAC, AND G. I. WEBB, *Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining*, Journal of Machine Learning Research, 10 (2009), pp. 377–403.

[135] M. OHSAKI, S. KITAGUCHI, K. OKAMOTO, H. YOKOI, AND T. YAMAGUCHI, *Evaluation of rule interestingness measures with a clinical dataset on hepatitis*, in PKDD, vol. 3202 of Lecture Notes in Computer Science, Springer, 2004, pp. 362–373.

[136] J. OUTRATA AND V. VYCHODIL, *Fast algorithm for computing fixpoints of galois connections induced by object-attribute relational data*, Inf. Sci., 185 (2012), pp. 114–127.

[137] O. OYSTEIN, *Galois connections*, Trans. Amer. Math. Soc, 55 (1944), pp. 494–513.

[138] B. ÖZDEN, S. RAMASWAMY, AND A. SILBERSCHATZ, *Cyclic association rules*, in ICDE, IEEE Computer Society, 1998, pp. 412–421.

[139] N. PASQUIER, Y. BASTIDE, R. TAOUIL, AND L. LAKHAL, *Discovering frequent closed itemsets for association rules*, in ICDT, vol. 1540 of Lecture Notes in Computer Science, Springer, 1999, pp. 398–416.

[140] Z. PAWLAK, *Rough sets*, International Journal of Parallel Programming, 11 (1982), pp. 341–356.

[141] G. PIATETSKY-SHAPIRO, *Discovery, analysis, and presentation of strong rules*, in Knowledge Discovery in Databases, AAAI/MIT Press, 1991, pp. 229–248.

[142] ——, *Knowledge discovery in real databases: A report on the IJCAI-89 workshop*, AI Magazine, 11 (1991), pp. 68–70.

[143] M. PLANTEVIT, A. LAURENT, D. LAURENT, M. TEISSEIRE, AND Y. W. CHOONG, *Mining multidimensional and multilevel sequential patterns*, TKDD, 4 (2010), pp. 4:1–4:37.

[144] H. A. PRIESTLEY, *Ordered sets and complete lattices*, in Algebraic and coalgebraic methods in the mathematics of program construction, Springer, 2002, pp. 21–78.

[145] L. QIN, J. X. YU, AND L. CHANG, *Diversifying top-k results*, PVLDB, 5 (2012), pp. 1124–1135.

[146] C. RAÏSSI, T. CALDERS, AND P. PONCELET, *Mining conjunctive sequential patterns*, Data Min. Knowl. Discov., 17 (2008), pp. 77–93.

[147] N. RAMAKRISHNAN, D. KUMAR, B. MISHRA, M. POTTS, AND R. F. HELM, *Turning cartwheels: an alternating algorithm for mining redescriptions*, in KDD, ACM, 2004, pp. 266–275.

[148] S. ROMAN, *Lattices and Ordered Sets*, Springer New York, 2008.

[149] B. S. SCHRÖDER, *Ordered sets*, Springer, 29 (2003), p. 30.

[150] A. SIEBES, *Data surveying: Foundations of an inductive query language*, in KDD, AAAI Press, 1995, pp. 269–274.

[151] A. SOULET AND F. RIOULT, *Efficiently depth-first minimal pattern mining*, in PAKDD (1), vol. 8443 of Lecture Notes in Computer Science, Springer, 2014, pp. 28–39.

[152] L. SZATHMARY, A. NAPOLI, AND P. VALTCHEV, *Towards rare itemset mining*, in ICTAI (1), IEEE Computer Society, 2007, pp. 305–312.

[153] L. SZATHMARY, P. VALTCHEV, A. NAPOLI, R. GODIN, A. BOC, AND V. MAKARENKOV, *A fast compound algorithm for mining generators, closed itemsets, and computing links between equivalence classes*, Ann. Math. Artif. Intell., 70 (2014), pp. 81–105.

[154] P. TAN, V. KUMAR, AND J. SRIVASTAVA, *Selecting the right interestingness measure for association patterns*, in KDD, ACM, 2002, pp. 32–41.

[155] R. E. TARJAN, *Depth-first search and linear graph algorithms*, SIAM J. Comput., 1 (1972), pp. 146–160.

[156] THE CGAL PROJECT, *CGAL User and Reference Manual*, CGAL Editorial Board, 4.9 ed., 2016.

[157] T. UNO, M. KIYOMI, AND H. ARIMURA, *LCM ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets*, in FIMI, vol. 126 of CEUR Workshop Proceedings, CEUR-WS.org, 2004.

[158] M. L. VAN DE VEL, *Theory of convex structures*, vol. 50, Elsevier, 1993.

[159] M. VAN LEEUWEN AND A. J. KNOBBE, *Non-redundant subgroup discovery in large and complex data*, in ECML/PKDD (3), vol. 6913 of Lecture Notes in Computer Science, Springer, 2011, pp. 459–474.

[160] ——, *Diverse subgroup set discovery*, Data Min. Knowl. Discov., 25 (2012), pp. 208–242.

[161] G. I. WEBB, *OPUS: an efficient admissible algorithm for unordered search*, J. Artif. Intell. Res., 3 (1995), pp. 431–465.

[162] R. WILLE, *Restructuring lattice theory: an approach based on hierarchies of concepts*, in Ordered sets, Springer, 1982, pp. 445–470.

[163] S. WROBEL, *An algorithm for multi-relational discovery of subgroups*, in PKDD, vol. 1263 of Lecture Notes in Computer Science, Springer, 1997, pp. 78–87.

[164] X. YAN AND J. HAN, *gspan: Graph-based substructure pattern mining*, in ICDM, IEEE Computer Society, 2002, pp. 721–724.

[165] ——, *Closegraph: mining closed frequent graph patterns*, in KDD, ACM, 2003, pp. 286–295.

[166] X. YAN, J. HAN, AND R. AFSHAR, *Clospan: Mining closed sequential patterns in large datasets*, in SDM, SIAM, 2003, pp. 166–177.

[167] Y. YANG, G. I. WEBB, AND X. WU, *Discretization methods*, in Data Mining and Knowledge Discovery Handbook, 2nd ed., Springer, 2010, pp. 101–116.

[168] M. J. ZAKI, *SPADE: an efficient algorithm for mining frequent sequences*, Machine Learning, 42 (2001), pp. 31–60.

[169] M. J. ZAKI AND M. OGIHARA, *Theoretical foundations of association rules*, in 3rd ACM SIGMOD workshop on research issues in data mining and knowledge discovery, 1998, pp. 71–78.

[170] B. ZALIK, *An efficient sweep-line Delaunay triangulation algorithm*, Computer-Aided Design, 37 (2005), pp. 1027–1038.

[171] T. ZHANG, *Association rules*, in PAKDD, vol. 1805 of Lecture Notes in Computer Science, Springer, 2000, pp. 245–256.

[172] S. ZILBERSTEIN, *Using anytime algorithms in intelligent systems*, AI Magazine, 17 (1996), pp. 73–83.

Abstract

As the title of this dissertation may suggest, the aim of this thesis is to provide an order-theoretic point of view on the task of *subgroup discovery*. Subgroup discovery is the automatic task of discovering interesting hypotheses in databases. That is, given a database, the hypothesis space the analyst wants to explore and a formal way of how the analyst gauges the quality of the hypotheses (e.g. a quality measure); the automated task of *subgroup discovery* aims to extract the interesting hypothesis w.r.t. these parameters. In order to elaborate fast and efficient algorithms for *subgroup discovery*, one should understand the underlying properties of the hypothesis space on the one hand and the properties of its quality measure on the other. In this thesis, we extend the state-of-the-art by: (i) providing a unified view of the hypotheses space behind subgroup discovery using the well-founded mathematical tool of *order theory*, (ii) proposing the new hypothesis space of conjunction of linear inequalities in numerical databases and the algorithms enumerating its elements and (iii) proposing an anytime algorithm for discriminative subgroup discovery on numerical datasets providing guarantees upon interruption.

Keywords: Knowledge Discovery in Databases, Subgroup Discovery, Discriminative Subgroup Discovery, Order Theory, Set Systems, Formal Concept Analysis, Pattern Setups

---

Résumé

Comme le titre pourrait le suggérer, l'objectif principal de cette thèse est de fournir une meilleure compréhension de la tâche de la découverte de sous-groupes à travers la théorie de l'ordre. La découverte de sous-groupes (Subgroup Discovery - SD) est la tâche automatique dont le but est la découverte d'hypothèses intéressantes dans les bases de données. Autrement dit, étant donnée une base de donnée, l'espace de recherche de toutes les hypothèses que l'analyste voudra tester ainsi qu'un moyen formel pour évaluer la qualité de ces hypothèses ; la tâche automatique de la découverte de sous-groupe s'efforce de trouver les meilleurs hypothèses quant à ces trois paramètres. Afin d'élaborer des algorithmes efficaces et efficients pour cette tâche, il est important de comprendre les propriétés des espaces de recherche d'une part et les propriétés de la mesure de qualité d'autre part. Dans cette thèse, nous étendons l'état de l'art par: (i) fournir une vue unifiée sur les espaces d'hypothèses derrière la tâche de découverte de sous-groupes en utilisant la théorie de l'ordre, (ii) proposer l'espace d'hypothèses de conjonctions d'inégalités linéaires dans les bases de données numériques ainsi que différents algorithmes permettant de les énumérer et (iii) proposer un algorithme anytime - fournit progressivement des résultats - pour la tâche particulière de fouille de sous-groupe discriminants dans les bases de données numériques. Ce dernier fournit des garanties sur la qualité des sous-groupes extraits même si l'algorithme est interrompu.

Mots-clés: Découverte de connaissances dans les données, Fouille de Sous-groupes, Fouille de Sous-groupes Discriminants, Théorie de l'ordre, Familles d'ensembles, Analyses de Concepts Formels, Contexte de Patrons

FOLIO ADMINISTRATIF

THESE DE L'UNIVERSITE DE LYON OPEREE AU SEIN DE L'INSA LYON

NOM : BELFODIL
(avec précision du nom de jeune fille, le cas échéant)

DATE de SOUTENANCE : 30/09/2019

Prénoms : Aimene

TITRE : An Order Theoritic Point-of-view on Subgroup Discovery

NATURE : Doctorat

Numéro d'ordre : 2019LYSEI078

Ecole doctorale : InfoMaths (ED 512)

Spécialité : Informatique

RESUME :
Comme le titre pourrait le suggérer, l'objectif principal de cette thèse est de fournir une meilleure compréhension de la tâche de la découverte de sous-groupes à travers la théorie de l'ordre. La découverte de sous-groupes (Subgroup Discovery - SD) est la tâche automatique dont le but est la découverte d'hypothèses intéressantes dans les bases de données. Autrement dit, étant donnée une base de donnée, l'espace de recherche de toutes les hypothèses que l'analyste voudra tester ainsi qu'un moyen formel pour évaluer la qualité de ces hypothèses ; la tâche automatique de la découverte de sous-groupe s'efforce de trouver les meilleurs hypothèses quant à ces trois paramètres. Afin d'élaborer des algorithmes efficaces et efficients pour cette tâche, il est important de comprendre les propriétés des espaces de recherche d'une part et les propriétés de la mesure de qualité d'autre part. Dans cette thèse, nous étendons l'état de l'art par: (i) fournir une vue unifiée sur les espaces d'hypothèses derrière la tâche de découverte de sous-groupes en utilisant la théorie de l'ordre, (ii) proposer l'espace d'hypothèses de conjonctions d'inégalités linéaires dans les bases de données numériques ainsi que différents algorithmes permettant de les énumérer et (iii) proposer un algorithme anytime - fournit progressivement des résultats - pour la tâche particulière de fouille de sous-groupe discriminants dans les bases de données numériques. Ce dernier fournit des garanties sur la qualité des sous-groupes extraits même si l'algorithme est interrompu.

MOTS-CLÉS : Découverte de connaissances dans les données, Fouille de Sous-groupes, Fouille de Sous-groupes Discriminants, Théorie de l'ordre, Familles d'ensembles, Analyses de Concepts Formels, Contexte de Patrons

Laboratoire (s) de recherche : Laboratoire d'InfoRmatique en Image et Systèmes d'information (LIRIS)

Directeur de thèse:
Pr. Céline Robardet (INSA de Lyon)
Dr. Mehdi Kaytoue (Infologic)

Président de jury :
Pr. Miguel Couceiro (Université de Lorraine)

Composition du jury :
Pr. Bruno Crémilleux (Université de Caen) - Rapporteur
Pr. Bernhard Ganter (Technische Universitaet Dresden) - Rapporteur
Dr. Peggy Cellier (INSA Rennes) - Examinatrice
Pr. Miguel Couceiro (Université de Lorraine) - Examinateur
Pr. Arno Siebes (Universiteit Utrecht) - Examinateur
Pr. Sergei O. Kuznetsov (Higher School of Economics - Moscow) - Invité
Mr. Julien Zarka (Mobile Devices Ingenierie) - Invité