ISIMA

Institut Supérieur d'Informatique, de Modélisation et de leurs Applications

1 rue de la Chebarde
Aubières, 63178, France

NATIONAL RESEARCH UNIVERSITY

Higher School of Economics

Kochnovskiy Proyezd, 3
Moscow, 125319, Russia

Master Thesis report:
Data science and 3rd year of computer science engineering

Horn Minimization: an overview of existing algorithms

Author : Simon Vilmin

Academic Supervisor : Sergei A. Obiedkov

Held : June, 26, 2018

Acknowledgement

List of Figures

List of Algorithms

Abstract

Contents

Introduction

Chapter 1

Introduction to implications through closure systems

In this first chapter, we will be involved in presenting our topic of minimization. For this ground to be understandable by as much readers as possible, we will heavily rely on toy examples to illustrate and provide intuition on the various notions we will introduce. To be more precise on the path we are about to follow in this chapter, we are first to expose an informal small example of the task we want to achieve. Then, we shall investigate the history of research on our topic, to act as an exposition of the actual knowledge on the question and to give a context to our study. For the rest of this chapter we will get familiar with mathematical objects called *closure operators* and *closure systems* modelling our problem. As we shall observe, the topic of minimization can be described in several mathematical frameworks. However, even if we describe briefly other objects in next chapters, we will stick to our closure framework in all the report in order to have a leading light among various different terminologies.

1.1 Implications and minimization: first meeting

Let us imagine we are some specialist of flowers and plants in general. As such, we are interested in studying *correlations* between plant characteristics. Some possible traits are: *colourful, bloom, wither, aquatic, seasonal, climbing, scented, flower, perennial* and so forth. Having observed countless plants during our studies, we are able to draw relations among all those *attributes*. For instance, we know that a plant having the attribute *flower* is likely to have traits *scent, bloom, wither* while a plant being *perennial* (i.e: does not need a lot of water to survive, like a cactus) is not likely to be *aquatic*.

Those relations *"if we have some attributes, we get those ones too"* depict correlation between attributes (not cause/consequence!). It is important to stress on the knowledge those relations bring. They just indicate that whenever we have say *flower*, we have also *colourful*. This is very different from saying that *because* some plant is a flower, it will be colourful. We call those correlation relations *implication* and use $flower \longrightarrow colourful$ to denote *"if we have the attribute flower, then we have colourful"*. Now let us give

some implications:

$$(colourful,\ bloom \longrightarrow seasonal),\ (colourful,\ wither \longrightarrow seasonal),\ (bloom \longrightarrow wither)$$

All those implications represent a certain amount of knowledge. While in our example they are not numerous we could imagine having tons of them. Hence we would wonder whether there is a way to reduce the number of implications while keeping all the knowledge they represent. This question is *minimization*. Actually, in our small example we can reduce the number of implications. Take *(colourful, bloom* $\longrightarrow$ *seasonal)*. We can derive this implication relation only with the two other ones. Indeed, because a plant *blooming* is likely to *wither* (3rd implication), we have *(colourful, bloom* $\longrightarrow$ *wither)*, but since we now have *wither* and *colourful* we also have *seasonal* (2nd implication). That is, the implication *(colourful, bloom* $\longrightarrow$ *seasonal)* is useless (or *redundant*) in our context and can be removed. Our set of implications will then be smaller, but pointing out the same relations as before.

To summarize, we have seen that out of a set of *attributes* we can draw several relations called *implications* providing some knowledge. We also realized that sometimes, some implications are not necessary. Consequently, the set of implications we are given can be *minimized* without altering the information it contains. This is the topic we were interested during this master thesis. In the next section, we will trace back the overhaul knowledge on this question.

1.2 Research on implications theories minimization

This section is intended to supply the reader with a general overview of the minimization topic. After a short contextual information, we focus on some relevant results on the question by providing references to algorithms and properties dedicated to our problem. Eventually, we situate our work within this context.

The question of minimization has been discussed and developed through various frameworks, and several computer scientists communities. Notice that in order not to make this synthesis too long, we will stay within the context of minimization and will not trace the field of implication theories in general. For a survey of this domain anyway, the reader should refer to [29]. Also, note that minimality in general terms is not unique. Indeed, one can define several type of minimality among implication systems. For instance, not only we can define minimality with respect to the number of implication within a system (which is our interest) but also with respect to the number of attributes in each implications. The former one is called *canonical* in relational database field, and *hyperarc minimum* within the graph context. Especially in the graph-theoretic and boolean logic settings, one can derive more types of minimality. For general introduction to boolean logic notations, we invite the reader to see [15]. In terms of propositional logic, implications are represented through Horn formulae. Interestingly, the minimization problem we are going to consider is the only one being polynomial time solvable. Other problems are proved to be NP-Complete or NP-Hard. For more discussion on other minimality definitions and their computational complexity, the reader should refer to [13, 6, 7, 5, 29, 11]. In particular for NP-Completeness in the canonical case, one can see [23]. In subsequent explanations, we will refer to minimization with respect to the number of implications.

To the best of our knowledge, the two first fields in which algorithms and properties of minimality arose are Formal Concept Analysis (FCA) (see [21, 20] for an introduction) and Database Theory (DB) (see [24]). Both sides were developed independently in the early 80's. For the first domain, characterization of minimality goes to Duquenne and Guigues [22], in which they describe the so-called *canonical basis* (also called *Duquenne-Guigues basis* after its authors) relying on the notion of pseudo-closed sets. For the database part, study of implications is made by Maier through FD's ([24, 17]). The polynomial time algorithm he gives for minimization heavily relies on a fast subroutine discovered by Beeri and Bernstein in [9], 1979.

From then on, knowledge increased over years and spread out over domains. Another algorithm based on a minimality theorem is given by Shock in 1986 ([25]). Unfortunately, as we shall see and as already discussed by Wild in [28] the algorithm may not be correct in general, even though the underlying theorem is. During the same period, Ausiello and al. brought the problem to graph-theoretic ground, and provided new structure known as *FD-Graph* and algorithm to represent and work on implication systems in [6, 4, 5]. This approach has been seen in graph theory as an extension of the transitive closure in graphs ([1]), but no consideration equivalent to minimization task seems to have been taken beforehand, as far as we know. Still in the 1980 decade, Ganter expressed the canonical basis formalized by Duquenne and Guigues in his paper related to algorithms in FCA, [20] through closure systems, pseudo-closed and quasi-closed sets. Next, Wild ([26, 27, 28]) linked within this set-theoretic framework both the relational databases, formal concept analysis and lattice-theoretic approach. In relating those fields, he describes an algorithm for minimizing a basis, similar to algorithms of Day and, somehow, Shock (resp. [18], [25]). This framework is the one we will use for our study, and can be found in more recent work by Ganter & Obiedkov in [7]. Also, the works of Maier and Duquenne-Guigues have been used in the lattice-theoretic context by Day in [18] to derive an algorithm based on congruence relations. For in-depth knowledge of implication system within lattice terminology, we can see [16] as an introduction and [10] for a survey. Later, Duquenne proposed some variations in Day's work with another algorithm in [19]. More recently, Boròs and al. by working in a boolean logic framework, exhibited a theorem on the size of canonical basis [12, 13]. They also gave a general theoretic approach that algorithm should do one way or another on reduction purpose. Out of these papers, Berczi & al. derived a new minimization procedure based on hypergraphs in [14]. Furthermore, an algorithm for computing the canonical basis starting from any system is given in [7].

Even though the work we are going to cite is not designed to answer this question of minimization, it must also be exposed as the algorithm is intimately related to DG basis and can be used for base reduction. The paper of Angluin and al. in query learning, see [2], provides an algorithm for learning a Horn representation of an unknown initial formula. It has been shown later by Ariàs and Alcazar ([3]) that the output of Angluin algorithm was always the Duquennes-Guigues basis.

Our purpose with this master thesis is to review and implement as much as possible the algorithms we exposed to provide a comparison. This comparison shall act as both theoretical and experimental statement of algorithm efficiency. As we already mentioned we will focus on closure theory framework. The reason for this choice is our starting point. Because we start from the algorithms provided by Wild and

because the closure framework is the one we are the most familiar with, we focus on clearly explain this terminology with examples. However, once we will be comfortable with those definitions, we will relate other frameworks to our main approach in the next chapter, to explain and draw parallels with other algorithms. In the next section we will focus on theoretical definitions we shall need to understand the algorithms we have implemented.

1.3 Implications and minimization: theoretic approach

Here we will dive into mathematical representation of the task we gave in the first section of this chapter. For the recall, our aim here is to get familiar with the representation being closest from closure systems. Most of the notions initially come from [22, 20, 27, 21] but the reader can also find more than sufficient explanations in [7, 29]. Readers with knowledge in relational databases will recognize most of functional dependency notations. The reason is close vicinity between implications and functional dependencies. Talking about our needs, we can consider them as equivalent notations. Actually, the real-life application our set up will be the closest from is FCA ([21]) as we shall see in the last chapter.

1.3.1 Implications and closure systems

The easiest object to project onto mathematical definitions is our attribute set. For all the report, we fix Σ to be a set of *attributes*. Usually, we will denote attributes by small letters: *a, b, c, ...* and subsets of Σ (groups of attributes) will be denoted by capital letters: *A, B, C, ...* We assume the reader to have few background in elementary set-theoretic notations.

Definition 1 (*Implication, implication system*)**.** *An implication over Σ is a pair (A, B) with $A, B \subseteq \Sigma$. It is usually denoted by $A \longrightarrow B$. A set $\mathcal{L}$ of implications is called an implication system, implication theory or implication(al) base(is).*

Note that given as is, this definition seems to lose the semantic relation we depicted earlier. But we should keep in mind that in our set up, we will be given implications more than an attribute set. Hence, implications will make sense on their own, independently from the attribute set they are drawn from. Quickly, remark that implications in logical terms are expressed as *Horn formulae* giving another of its names to implication theories. Also, in $A \longrightarrow B$, A is said to be the *premise* (or *body*) and B the *conclusion* (*head*).

Definition 2 (*Model*)**.** *Let $\mathcal{L}$ be an implication system over Σ, and $M \subseteq \Sigma$. Then:*

(i) M is a model of an implication $A \longrightarrow B$, written $M \models A \longrightarrow B$, if $B \subseteq M$ or $A \nsubseteq M$,

(ii) M is a model of $\mathcal{L}$ if $M \models A \longrightarrow B$ for all $A \longrightarrow B \in \mathcal{L}$.

The notion of model may seem disarming at first sight. But M being a model of $A \longrightarrow B$ simply means that, if A is included in M, then for the implication $A \longrightarrow B$ to hold in M, we must have B in M too. This still suits the intuitive notion of premise/conclusion. Placed in the context of M, $A \longrightarrow B$ says *"whenever we have A, we must also have B"*. Reader with some background in mathematical logic should be familiar with the notation $\models$, denoting semantic entailment, as opposed to $\vdash$ for syntactic deduction (see [15]). By a fortunate twist of fate, semantic entailment is our next step:

Definition 3 (*Semantic entailment*)**.** *We say that an implication $A \longrightarrow B$ semantically follows from $\mathcal{L}$, denoted $\mathcal{L} \models A \longrightarrow B$, if all models M of $\mathcal{L}$ are models of $A \longrightarrow B$.*

Because next definitions are going to be on a slightly different structure, even though closely related to implication systems of course, let us rest for a while and illustrate our definitions with an example.

Example Consider again our plant properties. Let $\Sigma = \{$*colourful, bloom, wither, seasonal, aquatic, perennial, flower, scented*$\}$. An implication could be *flower* $\longrightarrow$ *scented*, or even *(bloom, aquatic)* $\longrightarrow$ *colourful* if we get rid off semantic interpretations. An implication basis $\mathcal{L}$ is for instance:

(colourful, bloom $\longrightarrow$ *seasonal), (colourful, wither* $\longrightarrow$ *seasonal), (bloom* $\longrightarrow$ *wither)*

and $M =$*(colourful, bloom, seasonal)* is a model of *colourful, bloom* $\longrightarrow$ *seasonal* because both the head and the body of the implication belong to M. Also, M is not a model of $\mathcal{L}$ because it is not a model of *bloom* $\longrightarrow$ *wither*. A model of $\mathcal{L}$ could be *(bloom, wither)* or even the empty set $\emptyset$.

Next definitions are about closure operators, and closure systems. We need to ground ourselves in those definitions before returning to implications. 2^Σ is the set of all subsets of Σ, also named the *power set* of Σ.

Definition 4 (*Closure operator*)**.** *Let Σ be a set and $\phi : 2^\Sigma \longrightarrow 2^\Sigma$ an application on the power set of Σ. ϕ is a closure operator if $\forall X, Y \subseteq \Sigma$:*

(i) $X \subseteq \phi(X)$ *(extensive),*

(ii) $X \subseteq Y \longrightarrow \phi(X) \subseteq \phi(Y)$ *(monotone),*

(iii) $\phi(X) = \phi(\phi(X))$ *(idempotent).*

$X \subseteq \Sigma$ is called closed if $X = \phi(X)$.

Definition 5 (*Closure system*)**.** *Let Σ be a set, and $\Sigma^\phi \subseteq 2^\Sigma$. Σ^ϕ is called a closure system if:*

(i) $\Sigma \in \Sigma^\phi$,

(ii) if $\mathcal{S} \subseteq \Sigma^\phi$, *then* $\bigcap \mathcal{S} \in \Sigma^\phi$ *(closed under intersection).*

In the second definition, it is worth stressing on the fact that Σ^ϕ is a set of sets. Also, the notation Σ^ϕ may seem surprising, but it has been chosen purposefully. Indeed, to each closure system Σ^ϕ over Σ, we can associate a closure operator ϕ and vice-versa:

- from ϕ to Σ^ϕ: compute all closed sets of ϕ to obtain Σ^ϕ,
- from Σ^ϕ to ϕ: define $\phi(X)$ as the smallest element of Σ^ϕ (inclusion-wise) containing X. Observe that such a set always exists in Σ^ϕ because $\Sigma \in \Sigma^\phi$.

In any case, this notation used for clear exposition of the link between closure systems and closure operators will be adapted to our context of implication systems as we shall see later on. Notice that one can encounter another object, *closure space*, being a pair (Σ, ϕ) where Σ is a set and ϕ a closure operator over Σ. We are likely to find this notation notably in [26, 27] where a general theory of closure spaces is addressed.

Example Let us imagine we have four people: *Jezabel, Neige, Seraphin* and *Narcisse*. Let us assume they all know each other and then define a relation *"like"* between them. For instance, say *Séraphin likes Jezabel*. this relation is a *binary relation*: it relates pairs of elements. We can represent this relation by a graph where nodes are people and edges represent relations:

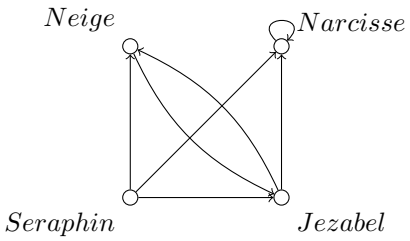


Figure 1.1: Graph of "like" relation

The arrow from *Seraphin* to *Jezabel* stands for *"Seraphin likes Jezabel"* and the arrow from *Narcisse* to itself means equivalently *"Narcisse likes Narcisse"*. With this clear, let us introduce an operation of gathering people. Starting from any group A of persons presented here, let's add to A every person liked by at least one element of A, until we can no more add people. For instance:

- if we start from *Neige*, because *Neige* likes *Jezabel* and *Jezabel* likes *Narcisse* we will add both of them to the group of *Neige*,
- because *Narcisse* only likes himself, we have no people to add in his group.

Now observe that this operation of gathering people is in fact a closure operator:

(i) it is *extensive*: starting from any group of people, we can only add new ones, hence either the group does not change (e.g: *Narcisse*) or it grows,

(ii) it is *monotone*: if we start from a group A containing a group B, it is clear that we will at least gather in A all the people we would add with B,

(iii) *idempotency*: once we added all the people we had to reach, then trying to find new people is useless by definition. Hence the group will remain the same if we apply our operation once more.

We are going to get back to our main implication purpose to illustrate the notion of closure in our context. It turns out that given a basis $\mathcal{L}$ over some set Σ, the set of models of $\mathcal{L}$, $\Sigma^{\mathcal{L}}$, is a closure system. Moreover, the operator $\mathcal{L} : 2^{\Sigma} \longrightarrow 2^{\Sigma}$ associating to a subset X of Σ the smallest model (inclusion wise) containing X is a closure operator. Furthermore, the closure system it defines is $\Sigma^{\mathcal{L}}$. An interesting point is the mathematical computation of $\mathcal{L}(X)$ given $\mathcal{L}$ as a set of implications. We rely on [26, 7] to this end. Let us define a temporary operation $\circ : 2^{\Sigma} \longrightarrow 2^{\Sigma}$ as follows:

$$X^{\circ} = X \cup \bigcup \{B \mid A \longrightarrow B \in \mathcal{L},\ A \subseteq X\}$$

Applying this operator up to stability provides $\mathcal{L}(X)$. In other words $\mathcal{L}(X) = X^{\circ\circ\cdots}$. It is clear that we have a finite amount of iterations since X cannot grow more than Σ. Readers with background in logic (see [13]) or graph theory ([14]) might see this operation as the marking or forward chaining procedure.

Example Let's stick to our vegetable example, but reducing Σ to {*bloom, flower, colourful* } (abbreviated *b, f, c*) for the sake of simplicity. Furthermore, let $\mathcal{L} = \{$*((colourful, bloom)* $\longrightarrow$ *flower), (flower* $\longrightarrow$ *bloom)*$\}$, abbreviated then $cb \longrightarrow f$, $f \longrightarrow b$. For instance, because $f \longrightarrow b \in \mathcal{L}$, the smallest model of $\mathcal{L}$ containing f is bf, and bf is closed. More precisely, the set of closed sets is the following:

$$\Sigma^{\mathcal{L}} = \{\emptyset,\ b,\ c,\ bf,\ bcf\}$$

Having presented the main definitions we shall need, we are to investigate practical computation of closures and more elaborated structures like the canonical basis (or Duquenne-Guigues basis) in the next section.

1.3.2 Canonical Basis, Closure Algorithm

Before giving the definition of canonical basis, we should consider special kind of sets given $\mathcal{L}$ over Σ. Also, we will need to expose particular implications. First of all, let us introduce a property through a proposition we will assume (we redirect the reader to [7] for another proof). When not introduced, we consider a system $\mathcal{L}$ of implications, over some attribute set Σ.

Proposition 1. *Let $A \longrightarrow B$ be an implication. $\mathcal{L} \models A \longrightarrow B$ if and only if $B \subseteq \mathcal{L}(A)$.*

Proof. $\mathcal{L} \models A \longrightarrow B \implies B \subseteq \mathcal{L}(A)$. Every model of $\mathcal{L}$ models $A \longrightarrow B$, hence for each closed set X of $\mathcal{L}$, either $A \subseteq X$ and $B \subseteq X$, or $A \nsubseteq X$. Consider all closed X for which $A \subseteq X$. By definition $\mathcal{L}(A) = \bigcap\{X \in \Sigma^{\mathcal{L}},\ A \subseteq X\}$ and $B \subseteq \mathcal{L}(A)$.

$B \subseteq \mathcal{L}(A) \implies \mathcal{L} \models A \longrightarrow B$. By contraposition suppose $\mathcal{L} \not\models A \longrightarrow B$. Then there must exist at least one model X of $\mathcal{L}$ such that $A \subseteq X$ and $B \nsubseteq X$. Because $A \subseteq X$, $\mathcal{L}(A) \subseteq X$ hence $B \nsubseteq \mathcal{L}(A)$.

□

Definition 6 (*Redundancy*)**.** *An implication $A \longrightarrow B$ of $\mathcal{L}$ is redundant if $\mathcal{L} - \{A \longrightarrow B\} \models A \longrightarrow B$. If $\mathcal{L}$ contains no redundant implications, it is non-redundant.*

Our definition of redundancy models the notion of *"useless"* we were talking about in our toy example: if an implication is true in some $\mathcal{L}$ even if we remove it, it brings no knowledge. In practice, redundancy can be checked as follows: put $\mathcal{L}^-$ as $\mathcal{L}$ without $A \longrightarrow B$ and compute $\mathcal{L}^-(A)$. If $\mathcal{L}^-(A) = \mathcal{L}(A)$ or equivalently, if $B \subseteq \mathcal{L}^-(A)$, then $A \longrightarrow B$ is redundant. Moreover, it is worth commenting that in FCA or DB fields (see [21, 24]), implications (or FD's) are deduced from data presented as contexts or relation schemes. Hence, we usually introduce notions of soundness and completeness ensuring that implications we are working on are meaningful with respect to the knowledge we are dealing with. More precisely, *soundness* ensures that $\mathcal{L}$ does not contain any implication not holding in the dataset. *Completeness* says that all true implications

in the data context are true in $\mathcal{L}$. Because we work directly on implications, $\mathcal{L}$ is by definition sound and complete with respect to the models it defines. Next, we set up minimality.

Definition 7 (Minimality)**.** *$\mathcal{L}$ is minimal if removing one of its implication alters $\Sigma^{\mathcal{L}}$.*

Example We consider our canonical plant example. Take

$$\mathcal{L} = \{((colourful,\ bloom) \longrightarrow seasonal),\ ((colourful,\ wither) \longrightarrow seasonal),\ (bloom \longrightarrow wither)\ \}$$

as we explained in first section, the first implication can be removed. In particular, it is redundant. Hence $\mathcal{L}$ is not minimal. If we get rid of $(colourful,\ bloom) \longrightarrow seasonal)$, $\mathcal{L}$ will be minimal.

Interestingly, depending on the implications we get, non-redundancy is not a sufficient criterion for minimality as we shall see in Maier algorithm. As an example for now, consider $\Sigma = \{a,\ b,\ c,\ d,\ e,\ f\}$ and $\mathcal{L} = \{ab \longrightarrow cde,\ c \longrightarrow a,\ d \longrightarrow b,\ cd \longrightarrow f\}$. $\mathcal{L}$ is not redundant, but is not minimal either. In fact, $\mathcal{L}_m = \{ab \longrightarrow cdef,\ c \longrightarrow a,\ d \longrightarrow b\}$ is equivalent to $\mathcal{L}$ but with one implication less.

For now, we defined what are implication theories, redundancy and minimality. One could expect our next step to be the exposition of some minimal basis. Unfortunately, we need to make a detour to visit some set and order definitions before getting back to our main purpose. Those notions not only deserve to explain minimal basis but also to settle some landmarks for further discussions in the next chapter.

Recall that in our example of closure operator we briefly approached binary relations. To be more formal, let E, F be two sets. A *binary relation* $\mathfrak{R}$ is a set of pairs (e, f) (sometimes denoted $e\mathfrak{R}f$) with $e \in E$, $f \in F$, or equivalently $\mathfrak{R} \subseteq E \times F$. We will assume $\mathfrak{R} \subseteq E^2$. Actually, $\mathfrak{R}$ can present some properties:

(i) *reflexivity*: $\forall x \in E, x\mathfrak{R}x$,

(ii) *irreflexivity*: $\forall x \in E, \neg(x\mathfrak{R}x)$,

(iii) *symmetry*: $\forall x, y \in E, x\mathfrak{R}y \longrightarrow y\mathfrak{R}x$

(iv) *antisymmetry*: $\forall x, y \in E, x\mathfrak{R}y \wedge y\mathfrak{R}x \longrightarrow x = y$,

(v) *asymmetry*: $\forall x, y \in E, x\mathfrak{R}y \longrightarrow \neg(y\mathfrak{R}x)$,

(vi) *transitivity*: $\forall x, y, z \in E, x\mathfrak{R}y \wedge y\mathfrak{R}z \longrightarrow x\mathfrak{R}z$

All possible properties are not given here, see [15] for more. With those properties anyway, we can define several types of relations:

Definition 8. *Let E be a set and $\mathfrak{R}$ a binary relation on E:*

(i) $\mathfrak{R}$ is an equivalence relation (denoted by $=$) if it is reflexive, transitive and symmetric,

(ii) $\mathfrak{R}$ is a (partial) order ($\leq$) if reflexive, transitive and antisymmetric,

(iii) $\mathfrak{R}$ is a strict order ($<$) if irreflexive, transitive and asymmetric.

Example Time has come for some illustrations. First, let us imagine we are looking at some tree in a meadow. Because the season is spring, this tree has branches and leaves. We are interested in the set of all leaves, and we would like to relate them by the branch they are one. Hence define $\mathfrak{R}$ as *"is on the same branch as"*, being a binary relation. It turns out that $\mathfrak{R}$ is an equivalence relation:

- *reflexivity*: every leaf is on the same branch as itself;
- *transitivity*: if a leaf l_1 is on the same branch as a leaf l_2, and l_2 is on the same branch as l_3, then it is clear that l_1 is on the same branch as l_3;
- *symmetry*: l_1 being on the same branch as l_2 clearly implies that l_2 is on the same branch as l_1.

For partial and strict ordering, we will go back to more mathematical examples, in order to slowly go back to our main purpose. Consider the set $\mathbb{N}$ $(= \mathbb{N}_0)$ of positive integers, including 0. The natural relation $\leq$ is an order, and the pair $(\mathbb{N}, \leq)$ is an ordered set. In particular it is a *totally ordered set* or *chain* because every pair of integers can be compared. $<$ is a strict total ordering on $\mathbb{N}$. Another example,let $\Sigma = \{a, b, c\}$ be a set of attributes and consider $\subseteq$ as a binary relation on 2^Σ. Again, $(2^\Sigma, \subseteq)$ is a *partially ordered set* (or *poset* under abbreviation):

- every subset X of Σ is included in itself, for instance $\{a, b\}$ is a subset or equal to $\{a, b\}$, whence *reflexivity*,
- if $X \subseteq Y$ and $Y \subseteq X$ then necessarily, $X = Y$ (*antisymmetry*),
- if $X \subseteq Y \subseteq Z$, then clearly $X \subseteq Z$ (*transitivity*)

There is a convenient way to represent posets. At least when they are not to heavy. It is sometimes called *Hasse diagram* (see [16]) and relies on the *cover* relation of a partially ordered set. Take any poset $(P, \leq)$ and define the cover relation as $x \prec y$ if $x < y$ and $x \leq z < y \longrightarrow x = z$. In other word, $x \prec y$ says *"there is no element between x and y"*. The example of $\mathbb{N}$ is appealing. For instance, $4 \prec 5$ because there is no integer between 4 and 5, but $4 \not\prec 7$ since we can find 5 and 6 as intermediary elements. Now the Hasse diagram of $(P, \leq)$ is a graph drawn as follows:

1. there is a point for each $x \in P$,
2. if $x \leq y$, then y is placed above x,
3. we draw an arc between x and y if and only if $x \prec y$ in P.

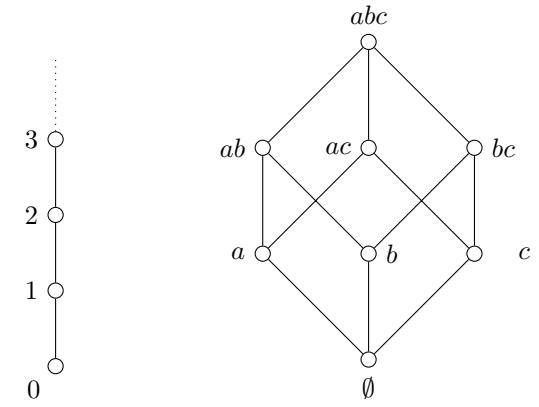
As examples, one can observe the diagrams of $(\mathbb{N}, \leq)$ and $(2^\Sigma, \subseteq)$ described previously in figure .

Definition 9 (*Pseudo-closed set*)**.** *Given $\mathcal{L}$ over Σ, we say that $P \subseteq \Sigma$ is pseudo-closed if:*

(i) $P \neq \mathcal{L}(P)$,

(ii) $Q \subset P$ *and* Q *pseudo-closed, implies* $\mathcal{L}(Q) \subseteq P$.

The idea of pseudo-closed sets goes back to Guigues and Duquenne in [22], but the name comes from Ganter in [20]. We can also find explanations in following research [20, 18] and in [7]. It turns out that we can explain pseudo-closure by using so called *quasi-closed sets* (see [26, 20, 22]).



(a) $(\mathbb{N}, \leq)$ (b) $(2^{\Sigma}, \subseteq)$

Figure 1.2: Hasse diagrams of two ordered sets

Definition 10 (*Quasi-closed set*)**.** *a set $Q \subseteq \Sigma$ is quasi-closed with respect to $\mathcal{L}$ if:*

(i) $Q \neq \mathcal{L}(Q)$,

(ii) $\forall A \subseteq Q$, $\mathcal{L}(A) \subseteq Q$ *or* $\mathcal{L}(A) = \mathcal{L}(Q)$.

Defining equivalence classes around closed sets, pseudo-closed sets are then minimal quasi-closed sets among their equivalence class. The interest of quasi-closure is that it can be seen out of the cube we are working on, as we shall see in next example.

Example Let us consider the following case:

- $\Sigma = \{a,\ b,\ c\}$,
- $\mathcal{L} = \{a \longrightarrow b,\ b \longrightarrow a,\ c \longrightarrow ab\}$.

As in **??**, we will represent the power set of Σ and equivalence classes of $\mathcal{L}$. Two subsets of Σ are in the same class if they have the same closure in $\mathcal{L}$. First, one can observe the effective class representation in figure 1.3. Models of $\mathcal{L}$ are indeed $\emptyset$, ab and abc. For instance $\mathcal{L}(ac) = abc$.

Next, we can observe figure 1.4 in which we somehow represented the definition of quasi-closure. We still represent equivalence classes. On the left-hand side figure, we consider the subset c. For c to be quasi-closed, we must look at all of its subsets, and see whether the closure of each subset is either smaller than c or in the same equivalence class as c. The dashed line shows which elements of the diagram we have to consider. In fact it represents what we call in lattice theory the *ideal* or *down-set* generated by c. It appears that the only distinct subset of c is $\emptyset$, which is closed. c itself is also not closed. Hence c is indeed quasi-closed.

On the right-hand side we consider the subset bc. As shown in the picture (under the dashed line), there are 3 elements to consider: $\emptyset$, c, b. For the same reason as for c, $\emptyset$ is not a problem. the closure of c is not included in bc, but equals the closure of bc. Hence c is not an issue either. However, b is included in bc, but

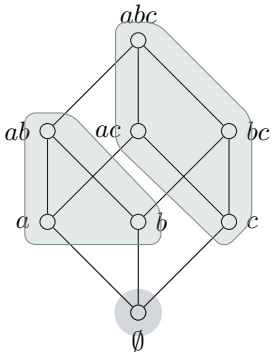
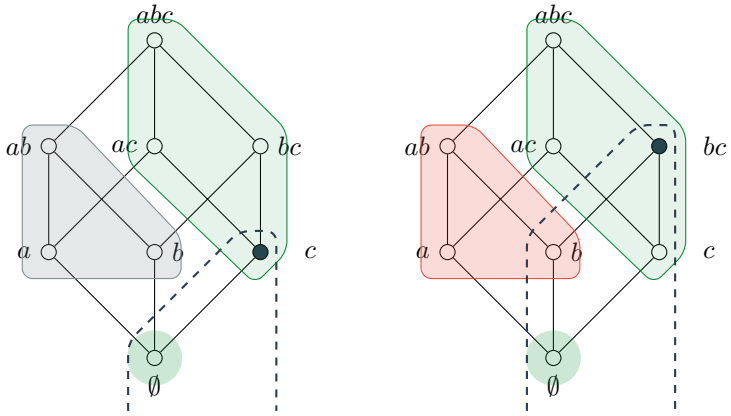


Figure 1.3: Equivalence classes representation of $\mathcal{L} = \{\ a \longleftrightarrow b,\ c \longrightarrow ab\ \}$



(a) Quasi-closed set: c (b) Non quasi-closed set: bc

Figure 1.4: Example of quasi-closeness in small implication system

its closure is ab, neither subset of bc nor equal to abc. Therefore, bc is not quasi-closed. Actually, one could informally say that a set Q is quasi-closed if the following is true:

$$(\forall P \in 2^{\Sigma})\,[(P \in \downarrow Q) \longrightarrow (\mathcal{L}(P) \in \downarrow Q \cup \{\mathcal{L}(Q)\})]$$

where $\downarrow Q$ is the ideal generated by Q (see dashed line in figure 1.4).

Example Now let us take

- $\Sigma = \{a,\ b,\ c\}$,
- $\mathcal{L} = \{c \longrightarrow ab,\ b \longrightarrow ab\}$.

Again, we will use equivalence classes and Hasse diagram to represent the closure system of $\mathcal{L}$, see figure 1.5. In this representation we coloured all quasi-closed sets at least in grey. Red (or lighter) nodes are precisely

pseudo-closed sets: they are the minimal quasi-closed sets among the equivalence class defined by *abc*.

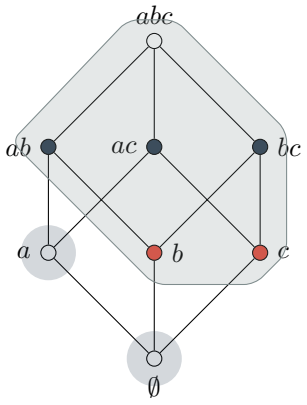


Figure 1.5: Pseudo-closed sets of $\mathcal{L} = \{ b \longrightarrow ac,\ c \longrightarrow ab \ \}$

Note that in particular, minimal premises of $\mathcal{L}$ inclusion wise are pseudo-closed. Furthermore, we should be aware that an equivalence class may not contain pseudo-closed set, or more generally, quasi-closed sets. As such, we cannot consider that minimal elements of equivalence classes are quasi-closed. Take for example $\mathcal{L} = \{ \emptyset \longrightarrow a,\ b \longrightarrow a \ \}$. b and ab define a class, but b is not even quasi-closed. With these notions, we can move on and define the canonical basis.

Definition 11 (*Duquenne-Guigues basis*)**.** *The basis $\mathcal{L}$ defined by*

$$\mathcal{L} = \{ P \longrightarrow \mathcal{L}(P) \mid P \text{ is pseudo-closed } \}$$

is called the Duquenne-Guigues or canonical basis. It is minimal.

This definition does not say that the canonical basis is the only one being minimal. Actually, it says that every minimal basis should have the same number of implications than this one. We can find a deeper argument in [7] on links between any minimal basis and the canonical one.

So far we discussed several notions: implications, pseudo-closed set, quasi-closed set, canonical basis and so forth. Most of them relies heavily on computing the closure of sets with respect to $\mathcal{L}$. Hence, to have practical efficiency, we must be able to compute closures as fast as possible. Fortunately, several algorithms can be found. Among them, there is a naïve procedure based on the operation $\circ$ we described earlier. Furthermore the algorithm by Beeri and Bernstein in [9] called LinClosure addresses this question. LinClosure as previously mentioned has been widely used, notably in [24, 17, 7, 25, 18]. Before describing those procedures, let us introduce our complexity notations:

- $|\Sigma|$ will denote the size of the attribute set Σ,
- $|\mathcal{B}|$ will be the number of implications in $\mathcal{L}$ ($\mathcal{B}$ stands for body),
- $|\mathcal{L}|$ is the number of symbols used to represent $\mathcal{L}$.

We consider $|\mathcal{L}|$ to be in reduced form for complexity results. By "reduced" we mean that we do not have distinct implications with same bodies. Indeed, if say, $a \longrightarrow b$ and $a \longrightarrow c$ holds in some $\Sigma^{\mathcal{L}}$ then we can replace those two implications by $a \longrightarrow bc$. Moreover, we shall not explain in details O notation for complexity since we do not need in-depth knowledge within this field. For us, it is enough to say that O is the asymptotically worst case complexity (in time or space). For instance, in the worst case, $|\mathcal{L}| = |\mathcal{B}| \times |\Sigma|$, thus $|\mathcal{L}| = O(|\mathcal{B}| \times |\Sigma|)$. Closure and LinClosure are algorithms 1, 2 (resp.).

Algorithm 1: Closure

Input: A base $\mathcal{L}$, $X \subseteq \Sigma$
Output: The closure $\mathcal{L}(X)$ of X under $\mathcal{L}$

```
closed := ⊥ ;
L(X) := X ;
while ¬closed do
    closed := ⊤ ;
    foreach A ⟶ B ∈ L do
        if A ⊆ L(X) then
            L(X) := L(X) ∪ B ;
            L := L − {A ⟶ B} ;
            closed := ⊥ ;
return L(X);
```

As we already mentioned, the algorithm Closure relies on the $\circ$ operation. The principle is to re-roll over the set of implications $\mathcal{L}$ to see whether there exists an implication $A \longrightarrow B$ in $\mathcal{L}$ such that $\mathcal{L}(X) \not\models A \longrightarrow B$ up to stability. Asymptotically, we will need $O(|\mathcal{B}|^2 \times |\Sigma|)$ if we remove only one implication per loop. the $|\Sigma|$ cost comes from the set union.

LinClosure has $O(|\mathcal{L}|)$ time complexity. The main idea is to use counters. Starting from X, if we reach for a given $A \longrightarrow B$ as many elements as $|A|$, then $A \subseteq \mathcal{L}(X)$ and we must also add B. Because the closure in itself is not the main point of our topic, we will not study LinClosure in depth. Furthermore, there exists other linear time algorithm for computing closure. For more complete theoretical and practical comparisons of closure algorithms, we redirect the reader to [8]. In this paper, LinClosure is shown maybe not to be the most efficient algorithm in practice when used in other algorithms, especially when compared with Closure. Anyway, because of its theoretical complexity and use in all algorithms we will review, we will still consider LinClosure, notably because we can separate the initialization step from the computation one in some cases on optimization purpose.

In this last section, we got a step further in building ground for understanding the implication theory structure. We gave definitions of minimality and visual examples of particular sets called pseudo-closed. With the support of those sets, we defined the canonical basis known to be minimal. Finally, algorithms for

Algorithm 2: LinClosure

Input: A base $\mathcal{L}$, $X \subseteq \Sigma$

Output: The closure $\mathcal{L}(X)$ of X under $\mathcal{L}$

```
foreach A ⟶ B ∈ L do
    count[A ⟶ B] := |A| ;
    if |A| = 0 then
        X := X ∪ B ;
    foreach a ∈ A do
        list[a] = list[a] ∪ {A ⟶ B} ;
update := X ;
while update ≠ ∅ do
    choose m ∈ update ;
    update := update − {m} ;
    foreach A ⟶ B ∈ list[m] do
        count[A ⟶ B] := count[A ⟶ B] − 1 ;
        if count[A ⟶ B] = 0 then
            add := B − X ;
            X := X ∪ add ;
            update := update ∪ add ;
return X ;
```

efficiently computing closures have been presented.

Conclusion In this chapter we first gave a soft introduction to our task with a somehow "physical" example. Then we described briefly advances starting from the first properties found independently in Concept Analysis and Relational Databases fields. We have seen that the question of Horn minimization has been studied in various fields such as graphs, closure spaces, logic (where the name Horn comes from), functional dependencies, lattices. Then, we placed our study within this context. The aim of this study has been exposed as providing a review of some algorithms we talked about, and comparing them under implementation. The last part of the chapter was dedicated to a more formal and theoretical ground necessary for a good understanding of subsequent parts. In the next chapter, we will theoretically discuss in details several algorithms.

Conclusion

Bibliography

[1] Aho, A. V., Garey, M. R., and Ullman, J. D. The Transitive Reduction of a Directed Graph. *SIAM Journal on Computing* (July 2006).

[2] Angluin, D., Frazier, M., and Pitt, L. Learning conjunctions of Horn clauses. *Machine Learning 9*, 2 (July 1992), 147–164.

[3] Arias, M., and Balcázar, J. L. Canonical Horn Representations and Query Learning. In *Algorithmic Learning Theory* (Berlin, Heidelberg, 2009), R. Gavaldà, G. Lugosi, T. Zeugmann, and S. Zilles, Eds., Springer Berlin Heidelberg, pp. 156–170.

[4] Ausiello, G., D'Atri, A., and Saccà, D. Graph Algorithms for Functional Dependency Manipulation. *J. ACM 30*, 4 (Oct. 1983), 752–766.

[5] Ausiello, G., D'Atri, A., and Saccá, D. Minimal Representation of Directed Hypergraphs. *SIAM J. Comput. 15*, 2 (May 1986), 418–431.

[6] Ausiello, G., and Laura, L. Directed hypergraphs: Introduction and fundamental algorithms—A survey. *Theoretical Computer Science 658*, Part B (2017), 293 – 306.

[7] B. Ganter, S. O. *Conceptual Exploration*. Springer, 2016.

[8] Bazhanov, K., and Obiedkov, S. Optimizations in computing the Duquenne–Guigues basis of implications. *Annals of Mathematics and Artificial Intelligence 70*, 1-2 (Feb. 2014), 5–24.

[9] Beeri, C., and Bernstein, P. A. Computational Problems Related to the Design of Normal Form Relational Schemas. *ACM Trans. Database Syst. 4*, 1 (Mar. 1979), 30–59.

[10] Bertet, K., Demko, C., Viaud, J.-F., and Guérin, C. Lattices, closures systems and implication bases: A survey of structural aspects and algorithms. *Theoretical Computer Science* (Nov. 2016).

[11] Boros, E., Čepek, O., and Kogan, A. Horn minimization by iterative decomposition. *Annals of Mathematics and Artificial Intelligence 23*, 3-4 (Nov. 1998), 321–343.

[12] Boros, E., Čepek, O., Kogan, A., and Kučera, P. Exclusive and essential sets of implicates of Boolean functions. *Discrete Applied Mathematics 158*, 2 (2010), 81 – 96.

[13] Boros, E., Čepek, O., and Makino, K. Strong Duality in Horn Minimization. In *Fundamentals of Computation Theory* (Berlin, Heidelberg, 2017), R. Klasing and M. Zeitoun, Eds., Springer Berlin Heidelberg, pp. 123–135.

[14] Bérczi, K., and Bérczi-Kovács, E. R. Directed hypergraphs and Horn minimization. *Information Processing Letters 128* (2017), 32 – 37.

[15] Cori, R., and Lascar, D. *Mathematical Logic: Part 1: Propositional Calculus, Boolean Algebras, Predicate Calculus, Completeness Theorems*. OUP Oxford, Sept. 2000. Google-Books-ID: Cle6_dOLt2IC.

[16] Davey, B. A., and Priestley, H. A. *Introduction to Lattices and Order*. Cambridge University Press, 2002.

[17] David, M. Minimum Covers in Relational Database Model. *J. ACM 27*, 4 (1980), 664 – 674.

[18] Day, A. The Lattice Theory of Functional Dependencies and Normal Decompositions. *International Journal of Algebra and Computation* (1992).

[19] Duquenne, V. Some variations on Alan Day's algorithm for calculating canonical basis of implications. In *Concept Lattices and their Applications (CLA)* (Montpellier, France, 2007), pp. 17–25.

[20] Ganter, B. Two Basic Algorithms in Concept Analysis. In *Formal Concept Analysis* (Mar. 2010), Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 312–340.

[21] Ganter, B., and Wille, R. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag, Berlin Heidelberg, 1999.

[22] Guigues J.L, D. V. Familles minimales d'implications informatives résultant d'un tableau de données binaires. *Mathématiques et Sciences Humaines 95* (1986), 5–18.

[23] Hammer, P. L., and Kogan, A. Optimal compression of propositional Horn knowledge bases: complexity and approximation. *Artificial Intelligence 64*, 1 (Nov. 1993), 131–145.

[24] Maier, D. *Theory of Relational Databases*. Computer Science Pr, 1983.

[25] Shock, R. C. Computing the minimum cover of functional dependencies. *Information Processing Letters 22*, 3 (Mar. 1986), 157–159.

[26] Wild, M. Implicational bases for finite closure systems. *Informatik-Bericht 89/3, Institut fuer Informatik* (Jan. 1989).

[27] Wild, M. A Theory of Finite Closure Spaces Based on Implications. *Advances in Mathematics 108*, 1 (Sept. 1994), 118–139.

[28] Wild, M. Computations with finite closure systems and implications. In *Computing and Combinatorics* (Aug. 1995), Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 111–120.

[29] Wild, M. The joy of implications, aka pure Horn formulas: Mainly a survey. *Theoretical Computer Science 658* (2017), 264 – 292.