Computing minimal generators from implications: a logic-guided approach

P. Cordero, M. Enciso, A. Mora, M Ojeda-Aciego*

Universidad de Málaga. Spain.
pcordero@uma.es, enciso@lcc.uma.es, amora@ctima.uma.es, aciego@uma.es

Abstract. Sets of attribute implications may have a certain degree of redundancy and the notion of basis appears as a way to characterize the implication set with less redundancy. The most widely accepted is the Duquenne-Guigues basis, strongly based on the notion of pseudo-intents. In this work we propose the minimal generators as an element to remove redundancy in the basis.
The main problem is to enumerate all the minimal generators from a set of implications. We introduce a method to compute all the minimal generators which is based on the Simplification Rule for implications. The simplification paradigm allows us to remove redundancy in the implications by deleting attributes inside the implication without removing the whole implication itself. In this work, the application of the Simplification Rule to the set of implications guides the search of the minimal generators in a logic-based style, providing a deterministic approach.

1 Introduction

A *concept* is a general idea that corresponds to some kind of entities and that may be characterized by some essential features of the class. When Wille [18] conceived Formal Concept Analysis (FCA), he probably did not foresee the wide diffusion of his original idea, both in the theoretical and in the applied areas. Application areas of FCA ranges from data analysis, information retrieval, or data mining to knowledge representation or the semantic web.

The main goal of Formal Concept Analysis is to identify the relationships between sets of objects and sets of attributes from information in a binary table. These relationships establish a Galois connection which allows us to identify the concepts by using lattice theory. The construction of the lattice of concepts is known to be a hard problem due to its exponential complexity. For this task, probably the most cited method is the NEXTCLOSURE algorithm developed by Ganter [5].

Apart from building the concept lattice itself, one of the key problems is to extract the set of attribute implications which hold in the concept lattice. In

* Supported by grants P09-FQM-5233 of the Junta de Andalucía, and TIN09-14562-C05-01 and TIN2011-28084 of the Science and Innovation Ministry of Spain, co-funded by the European Regional Development Fund (ERDF).

real applications the size of the concept lattice is usually huge and is impossible for the user to visualize it. Belohlavek and Vychodil developed [1] a method of expressing implications by means of closure operators. The main effect is to filter-out outputs to the user. The study of implications allowed those authors to reduce the number of formal concepts extracted from the input data: *"Using background knowledge thus enables a focused extraction of knowledge and may considerably reduce the amount of information presented to the user"* [2].

The problem arises when the set of implications has a high degree of redundancy: by the existence of redundant implications or redundant attributes inside the implications. One desired goal in this area is to remove redundancy and obtain a minimal basis. The most widely approach comes from the notion of *Duquenne-Guigues Basis* [6] also called *stem base*. This basis is minimal w.r.t. the number of implications, i.e. if one of the implications is removed from the basis, there are non-redundant implications which are valid in the dataset and cannot be inferred, using the Armstrong's Axioms, from the new reduced basis.

Nevertheless, this notion of minimality and its associated redundancy may be improved. To illustrate this assertion, we present here an example appeared in [5, pp. 30 and 84] where Ganter presents a context of developing countries. In the example, 130 countries and six attributes are considered: Group of 77, Non-aligned, LLDC (Least Developed Countries), MASC (Most Seriously Affected Countries), OPEC (Organization of Petrol Exporting Countries) and ACP (African, Caribbean and Pacific Countries). Ganter builds the concept lattice from this context and provides the following Duquenne-Guigues basis:

$$\begin{array}{rcl}
\text{OPEC} & \to & \text{Group 77, Non-aligned} \\
\text{MASC} & \to & \text{Group 77} \\
\text{Non-aligned} & \to & \text{Group of 77} \\
\text{Group 77, Non-aligned, MASC, OPEC} & \to & \text{LLDC, ACP} \\
\text{Group 77, Non-aligned, LLDC, OPEC} & \to & \text{MASC, ACP}
\end{array}$$

Note, however, that in the last two implications, there still exist redundant attributes in the left hand side, whereas in the first and in the last implications the redundancy appears in the right hand sides. This implies that it is possible to provide an equivalent and simpler set of implications:

$$\begin{array}{rcl}
\text{OPEC} & \to & \text{Non-aligned} \\
\text{MASC} & \to & \text{Group 77} \\
\text{Non-aligned} & \to & \text{Group of 77} \\
\text{MASC, OPEC} & \to & \text{LLDC, ACP} \\
\text{LLDC, OPEC} & \to & \text{MASC}
\end{array}$$

Thus, we have an example in which the Duquenne-Guigues basis still can contain redundant information. In order to obtain the latter basis it is necessary to consider minimal generators instead of pseudo-intents.

Implications define a closure system, and for this reason, a closure system can be replaced by their equivalent implications. The relationship between closed sets and implications are the key of the attribute exploration techniques [8, 13].

In [12] we introduced a closure algorithm strongly based on the simplification logic [3]. Here, our closure operator is used to guide the search of all the minimal generators of the closed sets corresponding to a given set of implications.

Methods for obtaining generators of closed sets have been studied in [4,10,17]. Minimal generators [14–16] appear in the literature under different names in various fields. For instance, in relational databases they are called minimal keys. In [17], the authors emphasize the importance of studying minimal generators although "they have been paid little attention so far in the FCA literature".

We would like to provide, in a further step, an alternative definition of basis built around the notion of minimal generator, since our goal is to avoid redundancy *inside* the implications, instead of just minimizing the number of implications. As a preliminary step, we have to design a method to enumerate all the minimal generators and their corresponding closed sets.

This paper is structured in the following way: in Section 2 some preliminaries about formal concept analysis and the Simplification paradigm are presented. In Section 3 a method to enumerate all the minimal generators from an implication set is introduced and in subsection 3.2 the algorithm is modified to compute the non-trivial minimal generators. The paper ends with a section on conclusions and prospects for future work.

2 Preliminaries

2.1 Formal Concept Analysis

Intuitively, Formal Concept Analysis (FCA) provides methods to describe the relationship between a set of objects and a set of attributes. A **formal context** is a triple $\mathbf{K} := (G, M, I)$ where G is a set of objects, M is a set of attributes and $I \subseteq G \times M$ is a binary relation between G and M such that, for $o \in G$ and $a \in M$, $o\ I\ a$ means that the object o has the attribute a. From this triple two mappings can be defined. One of them $(\ )' \colon 2^G \to 2^M$ is defined for all $A \subseteq G$ as $A' = \{m \in M \mid g\ I\ m \text{ for all } g \in A\}$. The other one, $(\ )' \colon 2^M \to 2^G$ is defined for all $B \subseteq M$ as $B' = \{g \in G \mid g\ I\ m \text{ for all } m \in B\}$. Both mappings are denoted by the same symbol because no confusion arises. This pair of mappings is a Galois connection and both are antitone mappings ($A_1 \subseteq A_2$ implies $A_2' \subseteq A_1'$ for all $A_1, A_2 \subseteq G$, and, for all $B_1, B_2 \subseteq M$, if $B_1 \subseteq B_2$ then $B_2' \subseteq B_1'$)

The composition of the intent and the extent mappings, and vice versa, give us two closure operators $(\ )'' \colon 2^G \to 2^G$ and $(\ )'' \colon 2^M \to 2^M$. That is, both are extensive ($X \subseteq X''$), idempotent ($(X'')'' = X''$) and isotone (if $X_1 \subseteq X_2$ then $X_1'' \subseteq X_2''$). In order to make this work self-contained, the notion of closed set (as a fixpoint of a closure operator) is defined below:

Definition 1. *A* ***formal concept*** *is a pair (A, B) such that $A \subseteq G$, $B \subseteq M$, $A' = B$ and $B' = A$. Consequently, A and B are closed sets of objects and attributes, respectively called extent and intent.*

It is well-known that the set of formal concepts is a complete lattice, the **concept lattice associated to the context**, with the following partial ordering is considered:

$$(A_1, B_1) \leq (A_2, B_2) \text{ if and only if } A_1 \subseteq A_2 \text{ (or equivalently } B_1 \supseteq B_2)$$

Related to the notion of closed set, the minimal generator (mingen) and implication are defined as follows:

Definition 2. *Let* $\mathbf{K} = (G, M, I)$ *be a formal context and* $A \subseteq M$*. The set of attributes* A *is said to be a minimal generator (**mingen**) if, for all set of attributes* $X \subseteq A$ *if* $X'' = A''$ *then* $X = A$*.*

Attribute implication allows us to capture all the information which can be deduced from a context. They are expressions $A \to B$ being A and B sets of attributes. A context satisfies the implication $A \to B$ if every object that has all the attributes from A also has all the attributes from B.

Definition 3. *An (attribute) implication of a formal context* $\mathbf{K} = (G, M, I)$ *is defined as a pair* (A, B)*, written* $A \to B$*, where* $A, B \subseteq M$*. Implication* $A \to B$ *holds (is valid) in* $\mathbf{K}$ *if* $A' \subseteq B'$*.*

The set of all valid implications in a context satisfies the well-known Armstrong's axioms:

$$\texttt{[Ref]}\ \frac{A \supseteq B}{A \to B}, \quad \texttt{[Augm]}\ \frac{A \to B}{A \cup C \to B \cup C}, \quad \texttt{[Trans]}\ \frac{A \to B,\ B \to C}{A \to C}$$

An implication **basis** of $\mathbf{K}$ is defined as a set $\mathcal{L}$ of implications of $\mathbf{K}$ from which any valid implication for $\mathbf{K}$ can be deduced by using Armstrong rules. The goal is to obtain a implication basis with minimal size. This condition is satisfied by the so-called Duquenne-Guigues basis [6] or stem basis, which is built over the set of the pseudo-intents [5].

As we have illustrated in the introduction, the definition of the Duquenne-Guigues basis refers to minimality only in the size of the set of implicants, but redundant attributes use to appear in this kind of minimal basis. This situation comes from the use of pseudo-intents in the construction of the basis. To avoid redundant attributes, we propose to consider minimal generators in the left hand side of the implications. In this work we do not provide such alternative definition of minimal basis, but focus on the problem of enumerating all the minimal generators. This is the main goal of Section 3, but before we need to recall the basics of the simplification logic.

2.2 Simplification logic and closures

Armstrong's axiom system has influenced the design of several logics for functional dependencies [9,11], all of them built around the transitivity paradigm. In this section, we summarize the axiom system of Simplification Logic for Functional Dependencies $\mathbf{SL}_{FD}$. It avoids the use of transitivity and is guided by the idea of simplifying the set of functional dependencies by efficiently removing some redundant attributes [3].

To begin with, $\mathbf{SL}_{FD}$ logic considers reflexivity as axiom scheme

$$\texttt{[Ref]}\quad \frac{A \supseteq B}{A \to B}$$

together with the following inference rules called fragmentation, composition and simplification respectively.

$$\texttt{[Frag]}\ \frac{A{\to}B\cup C}{A{\to}B}\qquad \texttt{[Comp]}\ \frac{A{\to}B,\ C{\to}D}{A\cup C{\to}B\cup D}\qquad \texttt{[Simp]}\ \frac{A{\to}B,\ C{\to}D}{A\cup(C\smallsetminus B){\to}D}$$

The equivalence between $\mathbf{SL}_{FD}$ logic and Armstrong's Axioms and an efficient algorithm to compute the closure of a set of attributes were proposed in [12]. We have also conducted a comparison to other closure algorithms in the literature in order to prove the better performance of our logic-based system. The main advantage of $\mathbf{SL}_{FD}$ is that inference rules can be considered as equivalence rules, and that is enough to compute all the derivations (see [12] for further details).

Proposition 1. *Let A, B, C, D sets of attributes. In $\mathbf{SL}_{FD}$ logic, the following equivalences hold:*

1. $\{A{\to}B\} \equiv \{A{\to}B\smallsetminus A\}$
2. $\{A{\to}B, A{\to}C\} \equiv \{A{\to}B\cup C\}$
3. $A\cap B=\varnothing$ *and* $A\subseteq C$ *imply* $\{A{\to}B, C{\to}D\} \equiv \{A{\to}B, C\smallsetminus B{\to}D\smallsetminus B\}$

It is well-known that, given a basis of a context, the closure of attribute sets defined based on Armstrong's axioms coincides with the closure $()'' : 2^M \to 2^M$. On the other hand, Armstrong's axioms and the axiom system in $\mathbf{SL}_{FD}$ logic are equivalent.

Definition 4. *Let Γ be a set of implications and A be a set of attributes. The closure of A in $\mathbf{SL}_{FD}$ logic is defined as the maximum set of attributes A^+ such that $\Gamma \vdash A{\to}A^+$.*

Theorem 1. *Let $\mathbf{K} = (G, M, I)$ a formal context and Γ a basis for $\mathbf{K}$. For all $A \subseteq M$, the equality $A^+ = A''$ holds.*

As we have said, in [12] we present a novel algorithm to compute closures using $\mathbf{SL}_{FD}$Logic. The formula $\varnothing{\to}A$ is used as a seed by the reasoning method to render the closure A^+ of A (which coincides with A'') just by applying some operators based on the [`Simp`] inference rule. The algorithm is based on the following results:

Theorem 2. *Let A and B be sets of attributes and Γ be a set of implications.*

$$\Gamma \vdash A{\to}B \quad \textit{if and only if} \quad \{\varnothing{\to}A\}\cup\Gamma \vdash \{\varnothing{\to}B\}$$

The closure algorithm is based on the previous theorem and the systematic application of the following equivalences which are direct consequences of the simplification equivalence (item 3 in Proposition 1).

Proposition 2. *Let A, B and C be sets of attributes, then the following equivalences hold:*

- **Eq. I**: *If $B \subseteq A$ then $\{\emptyset{\to}A, B{\to}C\} \equiv \{\emptyset{\to}A\cup C\}$.*
- **Eq. II**: *If $C \subseteq A$ then $\{\emptyset{\to}A, B{\to}C\} \equiv \{\emptyset{\to}A\}$.*
- **Eq. III**: *Otherwise $\{\emptyset{\to}A, B{\to}C\} \equiv \{\emptyset{\to}A, B\smallsetminus A{\to}C\smallsetminus A\}$.*

Given a set of implications Γ and a set of attributes A, the execution of the $\mathbf{SL}_{FD}$closure method begins with the construction of the guide $\emptyset{\to}A$. The three equivalences are systematically applied to Γ, which produces a growth of the guide. When none of the three equivalences can be applied, in the guide we have the closure.

Method 1 *Let Γ be a set of implications and A a subset of attributes. The following method computes the closure A^+ of the set A w.r.t. Γ:*

1. *Build the guide as follows $\{\emptyset{\to}A\}$*
2. *While there exist $B{\to}C \in \Gamma$ such that $A \cap B \neq \varnothing$ or $A \cap C \neq \varnothing$ (being $\{\varnothing{\to}A\}$ the guide in this state of the execution), execute the corresponding equivalence:*
   - *If $B \subseteq A$ then remove $B{\to}C$ from Γ and change the guide $\{\emptyset{\to}A\}$ by $\{\emptyset{\to}A \cup C\}$.*
   - *If $C \subseteq A$ then remove $B{\to}C$ from Γ.*
   - *Otherwise substitute $B{\to}C$ by $B \smallsetminus A{\to}C \smallsetminus A$ in Γ.*
3. *If $\{\varnothing{\to}A\}$ is the guide (in this state) then return A.*

Example 1. Let $\Gamma = \{ab{\to}c, bc{\to}d, de{\to}f, ce{\to}f\}$ and $A = de$. The following table summarizes the execution of Method 1 to compute $A^+ = def$.

Guide	Γ			
$\emptyset{\to}de$	$ab{\to}c$	$bc{\to}d$	$de{\to}f$	$ce{\to}f$
$\emptyset{\to}de$	$ab{\to}c$	$bc{\to}\cancel{d}$	$\cancel{d}\cancel{e}{\to}f$	$c\cancel{e}{\to}f$
$\emptyset{\to}def$	$ab{\to}c$			$c{\to}\cancel{f}$
$\emptyset{\to}def$	$ab{\to}c$			

The new closure operator was proven to be faster than other closure algorithms [12]. Finally, it is possible to define a modification of the closure algorithm to return not only the closure but also the resulting set Γ. This algorithm will be denoted by `Cls`. So, considering the previous example,

$$\texttt{Cls}(de, \{ab{\to}c, bc{\to}d, de{\to}f, ce{\to}f\}) = (def, \{ab{\to}c\})$$

3 Computing all mingens from a basis

In this section, we present how the Simplification paradigm can be considered as a powerful tool to guide the search of all minimal generator sets. In the first method we present here, we will compute mingens (the set of all minimal generators) from a set of implicant set. The algorithm works by applying the $\mathbf{SL}_{FD}$Closure algorithm to the set of implications. This application provides a new candidate to be added to mingen and a smaller implications set which guides us in the search of new sets of attributes to be added to mingens.

The input of this algorithm is a set of attributes M and a set of implications Γ over the attributes in M. The output is the set of closed sets endowed with all the mingens that generate them, i.e.

$$\{\langle C, mg(C)\rangle \colon C \text{ is a closed set of attributes}\}$$

where $mg(C) = \{D \colon D \text{ is a mingen and } D^+ = C\}$.

For example, if $M = \{a, b, c\}$ and $\Gamma = \{a \to b, b \to a\}$ the output must be

$$\{\langle abc, \{ac, bc\}\rangle, \langle ab, \{a, b\}\rangle, \langle c, \{c\}\rangle, \langle \varnothing, \{\varnothing\}\rangle\}$$

It can be seen as the concept lattice in which we have added labels with the mingens to each closed set.

With this idea we are going to need operators that allow us to work with this kind of sets, i.e. sets of pairs $\langle A, B\rangle$ such that $A \subseteq M$ is an intent and $B \subseteq 2^M$ satisfies the following conditions:

1. $X \subseteq A$ for all $X \in B$.
2. $X, Y \in B$ and $X \subseteq Y$ imply $X = Y$.

This kind of sets are called labeled set of intents (LSI). Given two LSIs Φ and Ψ, we define the join of both, $\Phi \sqcup \Psi$, as the the minimum LSI that satisfies:

– If $\langle A_1, B_1\rangle \in \Phi$ and $A_1 \neq A_2$ for all $\langle A_2, B_2\rangle \in \Psi$ then $\langle A_1, B_1\rangle \in \Phi \sqcup \Psi$
– If $\langle A_1, B_1\rangle \in \Psi$ and $A_1 \neq A_2$ for all $\langle A_2, B_2\rangle \in \Phi$ then $\langle A_1, B_1\rangle \in \Phi \sqcup \Psi$
– If $\langle A, B_1\rangle \in \Psi$ and $\langle A, B_2\rangle \in \Phi$ then $\langle A, B_3\rangle \in \Phi \sqcup \Psi$ being B_3 the set of minimal elements of $B_1 \cup B_2$.

We define an operator, the trivial operator, that given M and Γ returns the following LSI:

$$trv(M, \Gamma) = \{\langle X, \{X\}\rangle \colon X \subseteq M \text{ with } A \not\subseteq X \text{ for all } A \to B \in \Gamma\}$$

For example, $trv(\{a, b, c\}, \{a \to b, b \to a\}) = \{\langle c, \{c\}\rangle, \langle \varnothing, \{\varnothing\}\rangle\}$.

We will also need a way to add a pair to an LSI; it is defined as follows:

$$\texttt{Add}(\langle C, \{D\}\rangle, \Phi) = \{\langle A \cup C, \{X \cup D \colon X \in B\}\rangle \colon \langle A, B\rangle \in \Phi\}$$

For example,

$$\texttt{Add}(\langle ab, \{a\}\rangle, \{\langle c, \{c\}\rangle, \langle \varnothing, \{\varnothing\}\rangle\}) = \{\langle abc, \{ac\}\rangle, \langle ab, \{a\}\rangle\}$$
$$\texttt{Add}(\langle c, \{c\}\rangle, \{\langle ab, \{a, b\}\rangle, \langle \varnothing, \{\varnothing\}\rangle\}) = \{\langle abc, \{ac, bc\}\rangle, \langle c, \{c\}\rangle\}$$

3.1 MinGen Algorithm

We already have all the tools needed to define the algorithm, which is introduced below, together with an illustrative example of its application.

Example 2. We want to compute

$$\texttt{MinGen}(\{a, b, c, d\}, \{a \to b, c \to bd, bd \to ac\})$$

<u>Step 1.</u>
$$\begin{aligned}\Phi &= trv(\{a, b, c, d\}, \{a \to b, c \to bd, bd \to ac\} = \\ &= \{\langle b, \{b\}\rangle, \langle d, \{d\}\rangle, \langle \varnothing, \{\varnothing\}\rangle\}\end{aligned}$$

Algorithm 1: MinGen

```
Data: $M, \Gamma$
Result: $\Phi$
begin
    if $\Gamma = \varnothing$ then
        $\Phi = trv(M, \Gamma)$
    else
        Let $\Phi = trv(M, \Gamma)$;
        foreach $A \to B \in \Gamma$ do
            Let $(A^+, \Gamma') = \texttt{Cls}(A, \Gamma)$;
            Let $\Phi := \Phi \sqcup \texttt{Add}(\langle A^+, \{A\}\rangle, \texttt{MinGen}(M \smallsetminus A^+, \Gamma')$;
    Return $\Phi$
```

Step 2. Considering $a{\to}b \in \Gamma$,

$$\texttt{Cls}(a, \Gamma) = (ab, \{c{\to}d, d{\to}c\})$$

$\emptyset{\to}a$	$a{\to}b$	$c{\to}bd$	$bd{\to}ac$
$\emptyset{\to}a$	$\cancel{a}{\to}b$	$c{\to}bd$	$bd{\to}\cancel{a}c$
$\emptyset{\to}ab$		$c{\to}\cancel{b}d$	$\cancel{b}d{\to}c$

and call to $\texttt{MinGen}(\{c, d\}, \{c{\to}d, d{\to}c\})$

Step 2.1. This label (2.1) points that we have passed to a lower level, i.e. we have made a recursive call to the procedure.

$$\Phi_1 = trv(\{c, d\}, \{c{\to}d, d{\to}c\} = \{\langle\varnothing, \{\varnothing\}\rangle\}$$

Step 2.2. Considering $c{\to}d \in \Gamma_1$,

$$\texttt{Cls}(c, \Gamma_1) = (cd, \varnothing)$$

$\emptyset{\to}c$	$c{\to}d$	$d{\to}c$
$\emptyset{\to}c$	$\cancel{c}{\to}d$	$d{\to}\cancel{c}$
$\emptyset{\to}cd$		

and $\texttt{MinGen}(\varnothing, \varnothing) = \{\langle\varnothing, \{\varnothing\}\rangle\}$.

And so

$$\begin{aligned}\Phi_1 &= \{\langle\varnothing, \{\varnothing\}\rangle\} \sqcup \texttt{Add}(\langle cd, \{c\}\rangle, \{\langle\varnothing, \{\varnothing\}\rangle\}) \\ &= \{\langle cd, \{c\}\rangle, \langle\varnothing, \{\varnothing\}\rangle\}\end{aligned}$$

Step 2.3. Considering $d{\to}c \in \Gamma_1$,

$$\texttt{Cls}(d, \Gamma_1) = (cd, \varnothing)$$

$\emptyset{\to}d$	$c{\to}d$	$d{\to}c$
$\emptyset{\to}d$	$c{\to}\cancel{d}$	$\cancel{d}{\to}c$
$\emptyset{\to}cd$		

and $\texttt{MinGen}(\varnothing, \varnothing) = \{\langle\varnothing, \{\varnothing\}\rangle\}$.

And so

$$\begin{aligned}\Phi_1 &= \{\langle cd, \{c\}\rangle, \langle\varnothing, \{\varnothing\}\rangle\} \sqcup \texttt{Add}(\langle cd, \{d\}\rangle, \{\langle\varnothing, \{\varnothing\}\rangle\}) \\ &= \{\langle cd, \{c, d\}\rangle, \langle\varnothing, \{\varnothing\}\rangle\}\end{aligned}$$

Returning to the higher level,

$$\begin{aligned}\Phi &= \{\langle b, \{b\}\rangle, \langle d, \{d\}\rangle, \langle\varnothing, \{\varnothing\}\rangle\} \sqcup\ \texttt{Add}(\langle ab, \{a\}\rangle, \{\langle cd, \{c, d\}\rangle, \langle\varnothing, \{\varnothing\}\rangle\}) \\ &= \{\langle abcd, \{ac, ad\}\rangle, \langle ab, \{a\}\rangle, \langle b, \{b\}\rangle, \langle d, \{d\}\rangle, \langle\varnothing, \{\varnothing\}\rangle\}\end{aligned}$$

Step 3. Considering $c{\to}bd \in \Gamma$,

$$\texttt{Cls}(bc, \Gamma) = (abcd, \varnothing)$$

$\emptyset{\to}c$	$a{\to}b$	$c{\to}bd$	$bd{\to}ac$
$\emptyset{\to}c$	$a{\to}b$	$\cancel{c}{\to}bd$	$bd{\to}a\cancel{c}$
$\emptyset{\to}bcd$	$a{\to}\cancel{b}$		$\cancel{b}\cancel{d}{\to}a$
$\emptyset{\to}abcd$			

and $\texttt{MinGen}(\varnothing, \varnothing) = \{\langle\varnothing, \{\varnothing\}\rangle\}$.

$$\begin{aligned}\Phi &= \{\langle abcd, \{ac, ad\}\rangle, \langle ab, \{a\}\rangle\langle b, \{b\}\rangle, \langle d, \{d\}\rangle, \langle\varnothing, \{\varnothing\}\rangle\} \\ &\sqcup \texttt{Add}(\langle abcd, \{c\}\rangle, \{\langle\varnothing, \{\varnothing\}\rangle\}) \\ &= \{\langle abcd, \{c, ad\}\rangle, \langle ab, \{a\}\rangle, \langle b, \{b\}\rangle, \langle d, \{d\}\rangle, \langle\varnothing, \{\varnothing\}\rangle\}\end{aligned}$$

Note that $\{\langle abcd, \{ac, ad\}\rangle\} \sqcup \{\langle abcd, \{c\}\rangle\} = \{\langle abcd, \{c, ad\}\rangle\}$.

Step 4. Considering $bd{\to}ac \in \Gamma$,

$$\texttt{Cls}(bd, \Gamma) = (abcd, \varnothing)$$

and $\texttt{MinGen}(\varnothing, \varnothing) = \{\langle \varnothing, \{\varnothing\}\rangle\}$.

$\emptyset{\to}bd$	$a{\to}b$	$c{\to}bd$	$bd{\to}ac$
$\emptyset{\to}bd$	$a{\to}\cancel{b}$	$c{\to}\cancel{b}\cancel{d}$	$\cancel{b}\cancel{d}{\to}ac$
$\emptyset{\to}abcd$			

$$\begin{aligned}\Phi &= \{\langle abcd, \{c, ad\}\rangle, \langle ab, \{a\}\rangle, \langle b, \{b\}\rangle, \langle d, \{d\}\rangle, \langle \varnothing, \{\varnothing\}\rangle\}\\ &\sqcup \texttt{Add}(\langle abcd, \{bd\}\rangle, \{\langle \varnothing, \{\varnothing\}\rangle\})\\ &= \{\langle abcd, \{c, ad, bd\}\rangle, \langle ab, \{a\}\rangle \langle b, \{b\}\rangle, \langle d, \{d\}\rangle, \langle \varnothing, \{\varnothing\}\rangle\}\end{aligned}$$

Finally, the algorithm returns

$$\{\langle abcd, \{c, ad, bd\}\rangle, \langle ab, \{a\}\rangle \langle b, \{b\}\rangle, \langle d, \{d\}\rangle, \langle \varnothing, \{\varnothing\}\rangle\}$$

3.2 Non-trivial minimal generators.

Notice that the first method we have just introduced computes all minimal generators, including trivial mingens with non-relevant information.

In this subsection, we slightly modify the previous algoritm to produce only the relevant information (not trivial mingens), by considering that $trv(M, \Gamma)$ returns always only $\{\langle \varnothing, \{\varnothing\}\rangle\}$. This new algorithm is called L $\texttt{MinGen}_0$.

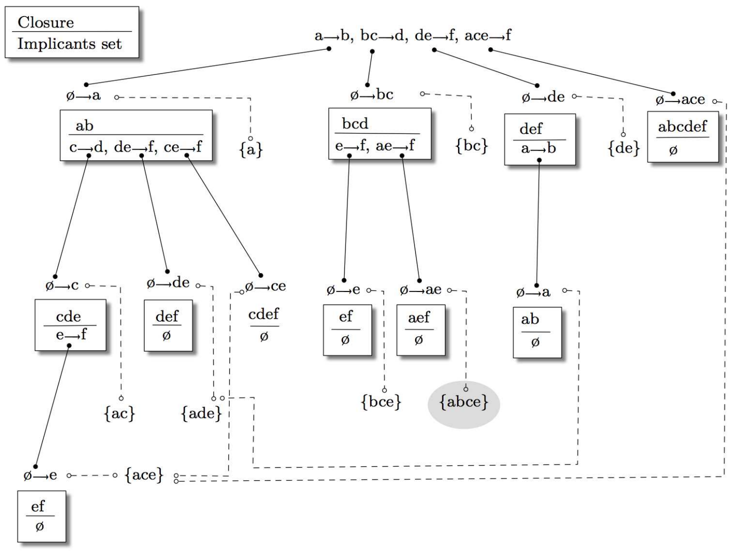

Fig. 1. SL$_{FD}$ guides the construction of the MinGen set

Example 3. We want L to compute

$$\texttt{MinGen}_0(\{a,b,c,d,e,f\},\{a\to b, bc\to d, de\to f, ace\to f\})$$

The full execution of the method $\texttt{MinGen}_0$ is depicted in Figure 1. In Figure 2 we show the lattice of closed sets built with the non-trivial mingens provided by $\texttt{MinGen}_0$.

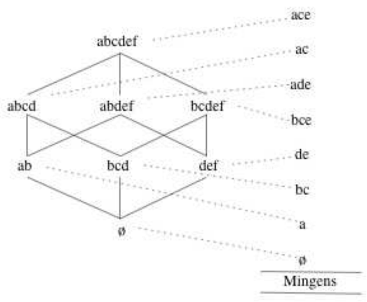

Fig. 2. Lattice of minimal generators

Step 1. $\Phi = \{\langle\varnothing,\{\varnothing\}\rangle\}$
Step 2. Considering $a\to b\in\Gamma$, compute

$$\texttt{Cls}(a,\Gamma) = (ab,\{c\to d, de\to f, ce\to f\})$$

Now we compute $\texttt{MinGen}_0(\{c,d,e,f\},\{c\to d, de\to f, ce\to f\})$
Step 2.1. $\Phi_1 = \{\langle\varnothing,\{\varnothing\}\rangle\}$
Step 2.2. Considering $c\to d\in\Gamma'$, compute $\texttt{Cls}(c,\Gamma) = (cd,\{e\to f\})$. Now we are going to compute $\texttt{MinGen}_0(\{e,f\},\{e\to f\})$.
Step 2.2.1 $\Phi_{1.1} = \{\langle\varnothing,\{\varnothing\}\rangle\}$
Step 2.2.2 Considering $e\to f\in\Gamma''$, $\texttt{Cls}(e,\Gamma'') = (ef,\varnothing)$. And so

$$\begin{aligned}\Phi_{1.1} &= \{\langle\varnothing,\{\varnothing\}\rangle\}\sqcup\texttt{Add}(\langle ef,\{e\}\rangle,\{\langle\varnothing,\{\varnothing\}\rangle\})\\ &= \{\langle ef,\{e\}\rangle,\langle\varnothing,\{\varnothing\}\rangle\}\end{aligned}$$

Returning to the higher level,

$$\begin{aligned}\Phi_1 &= \{\langle\varnothing,\{\varnothing\}\rangle\}\sqcup\texttt{Add}(\langle cd,\{c\}\rangle,\{\langle ef,\{e\}\rangle,\langle\varnothing,\{\varnothing\}\rangle\})\\ &= \{\langle cdef,\{ce\}\rangle,\langle cd,\{c\}\rangle,\langle\varnothing,\{\varnothing\}\rangle\}\end{aligned}$$

Step 2.3. Considering $de\to f\in\Gamma'$, $\texttt{Cls}(de,\Gamma) = (def,\varnothing)$. Moreover, $\texttt{MinGen}_0(\{c\},\varnothing) = \{\langle\varnothing,\{\varnothing\}\rangle\}$

$$\begin{aligned}\Phi_1 &= \{\langle cdef,\{ce\}\rangle,\langle cd,\{c\}\rangle,\langle\varnothing,\{\varnothing\}\rangle\}\sqcup\texttt{Add}(\langle def,\{de\}\rangle,\{\langle\varnothing,\{\varnothing\}\rangle\})\\ &= \{\langle cdef,\{ce\}\rangle,\langle def,\{de\}\rangle,\langle cd,\{c\}\rangle,\langle\varnothing,\{\varnothing\}\rangle\}\end{aligned}$$

Step 2.4. Considering $ce\to f\in\Gamma'$, compute $\texttt{Cls}(ce,\Gamma) = (cdef,\varnothing)$. Moreover $\texttt{MinGen}_0(\varnothing,\varnothing) = \{\langle\varnothing,\{\varnothing\}\rangle\}$

$$\begin{aligned}\Phi_1 &= \{\langle cdef,\{ce\}\rangle,\langle def,\{de\}\rangle,\langle cd,\{c\}\rangle,\langle\varnothing,\{\varnothing\}\rangle\}\sqcup\texttt{Add}(\langle cdef,\{ce\}\rangle,\{\langle\varnothing,\{\varnothing\}\rangle\})\\ &= \{\langle cdef,\{ce\}\rangle,\langle def,\{de\}\rangle,\langle cd,\{c\}\rangle,\langle\varnothing,\{\varnothing\}\rangle\}\end{aligned}$$

Returning to the high level,

$$\begin{aligned}\Phi &= \{\langle\varnothing,\{\varnothing\}\rangle\} \sqcup \texttt{Add}(\langle ab,\{a\}\rangle,\{\langle cdef,\{ce\}\rangle,\langle def,\{de\}\rangle,\langle cd,\{c\}\rangle,\langle\varnothing,\{\varnothing\}\rangle\})\\ &= \{\langle abcdef,\{ace\}\rangle,\langle abdef,\{ade\}\rangle,\langle abcd,\{ac\}\rangle,\langle ab,\{a\}\rangle,\langle\varnothing,\{\varnothing\}\rangle\}\end{aligned}$$

Step 3. Considering $bc{\to}d \in \Gamma$, $\texttt{Cls}(bc,\Gamma) = (bcd,\{e{\to}f, ae{\to}f\})$. Now we compute $\texttt{MinGen}_0(\{a,e,f\},\{e{\to}f, ae{\to}f\})$.
Step 3.1. $\Phi_2 = \{\langle\varnothing,\{\varnothing\}\rangle\}$
Step 3.2. Considering $e{\to}f \in \Gamma'$, compute $\texttt{Cls}(e,\Gamma) = (ef,\varnothing)$ and $\texttt{MinGen}_0(\{a\},\varnothing) = \{\langle\varnothing,\{\varnothing\}\rangle\}$

$$\Phi_2 = \{\langle\varnothing,\{\varnothing\}\rangle\} \sqcup \texttt{Add}(\langle ef,\{e\}\rangle,\{\langle\varnothing,\{\varnothing\}\rangle\}) = \{\langle ef,\{e\}\rangle,\langle\varnothing,\{\varnothing\}\rangle\}$$

Step 3.3. Considering $ae{\to}f \in \Gamma'$, $\texttt{Cls}(ae,\Gamma) = (aef,\varnothing)$. Moreover $\texttt{MinGen}_0(\varnothing,\varnothing) = \{\langle\varnothing,\{\varnothing\}\rangle\}$

$$\begin{aligned}\Phi_2 &= \{\langle ef,\{e\}\rangle,\langle\varnothing,\{\varnothing\}\rangle\} \sqcup \texttt{Add}(\langle aef,\{ae\}\rangle,\{\langle\varnothing,\{\varnothing\}\rangle\})\\ &= \{\langle aef,\{ae\}\rangle,\langle ef,\{e\}\rangle,\langle\varnothing,\{\varnothing\}\rangle\}\end{aligned}$$

Returning to the high level,

$$\begin{aligned}\Phi &= \{\langle abcdef,\{ace\}\rangle,\langle abdef,\{ade\}\rangle,\langle abcd,\{ac\}\rangle,\langle ab,\{a\}\rangle,\langle\varnothing,\{\varnothing\}\rangle\}\\ &\quad\sqcup\texttt{Add}(\langle bcd,\{bc\}\rangle,\{\langle aef,\{ae\}\rangle,\langle ef,\{e\}\rangle,\langle\varnothing,\{\varnothing\}\rangle\})\\ &= \{\langle abcdef,\{ace\}\rangle,\langle abdef,\{ade\}\rangle,\langle abcd,\{ac\}\rangle,\langle ab,\{a\}\rangle,\langle\varnothing,\{\varnothing\}\rangle\}\\ &\quad\sqcup\{\langle abcdef,\{abce\}\rangle,\langle bcdef,\{bce\}\rangle,\langle bcd,\{bc\}\rangle\}\\ &=\{\langle abcdef,\{ace\}\rangle,\langle abdef,\{ade\}\rangle,\langle bcdef,\{bce\}\rangle,\langle abcd,\{ac\}\rangle,\\ &\quad\langle bcd,\{bc\}\rangle,\langle ab,\{a\}\rangle,\langle\varnothing,\{\varnothing\}\rangle\}\end{aligned}$$

Notice that the last application of the $\sqcup$ operator does not add the set $\{abce\}$ to mingen because it produces the same closed set than $\{ace\}$. Thus, in Figure 1 this leaf appears with gray color.
Step 4. Considering $de{\to}f \in \Gamma$, renders

$$\begin{aligned}\Phi = \{&\langle abcdef,\{ace\}\rangle,\langle abdef,\{ade\}\rangle,\langle bcdef,\{bce\}\rangle,\langle abcd,\{ac\}\rangle,\\ &\langle bcd,\{bc\}\rangle,\langle def,\{de\}\rangle,\langle ab,\{a\}\rangle,\langle\varnothing,\{\varnothing\}\rangle\}\end{aligned}$$

Step 5. Considering $ace{\to}f \in \Gamma$, renders

$$\begin{aligned}\Phi = \{&\langle abcdef,\{ace\}\rangle,\langle abdef,\{ade\}\rangle,\langle bcdef,\{bce\}\rangle,\langle abcd,\{ac\}\rangle,\\ &\langle bcd,\{bc\}\rangle,\langle def,\{de\}\rangle,\langle ab,\{a\}\rangle,\langle\varnothing,\{\varnothing\}\rangle\}\end{aligned}$$

4 Conclusions and future works

In this work we have proposed the minimal generators as a basic notion to remove redundancy in sets of implications. To achieve this goal, the first step is to design a method to compute all minimal generators corresponding to a set of implications. The simplification paradigm is the key to design the `MinGen` and $\texttt{MinGen}_0$ algorithms. The application of $\mathbf{SL}_{FD}$closure algorithm guides the search of minimal generators.

As future work we will present a thorough study about the soundness, completeness, and complexity of the mingens algorithms, which have not been included here due to space limitations; moreover, a definition of basis center within the notion of minimal generators will be studied, together with a method to compute such a basis.

References

1. R. Belohlavek and V. Vychodil, Formal Concept Analysis with Constraints by Closure Operators, LNCS, 4068: 131–143, 2006.
2. R. Belohlavek and V. Vychodil, Formal Concept Analysis with Background Knowledge: Attribute Priorities. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 39: 399–409, 2009.
3. P. Cordero, M. Enciso, A. Mora, I.P, de Guzmán: SLFD logic: Elimination of data redundancy in knowledge representation, LNCS 2527: 141–150, 2002.
4. C. Frambourg, P. Valtchev, and R. Godin, Merge-based computation of minimal generators. Discrete Mathematics, LNCS 3596: 181–194, 2005.
5. B. Ganter, Two basic algorithms in concept analysis. Technische Hochschule, Darmstadt, 1984.
6. J.L. Guigues and V. Duquenne, Familles minimales d'implications informatives résultant d'un tableau de données binaires. Math. Sci. Humaines, 95, 5–18, 1986.
7. M. Hermann and B. Sertkaya, On the Complexity of Computing Generators of Closed Sets. ICFCA 2008: 158–168
8. S. O. Kuznetsov and A. Revenko, Attribute Exploration of Properties of Functions on Sets. Fundamenta Informaticae, 115 (4): 377–394, 2012.
9. C. Beeri, and R. Fagin and J.H. Howard, *A complete axiomatization for functional and multivalued dependencies in database relations*, Proc. ACM SIGMOD Conf., pp. 47-61, 1977.
10. A. Day, *The lattice theory of functional dependencies and normal decompositions*, International Journal of Algebra and Computation, 2: pp. 209–431, 1990.
11. R. Fagin, *Functional dependencies in a relational database and propositional logic*, IBM. Journal of research and development, 21 (6), (1977), pp. 534–544.
12. A. Mora, P. Cordero, M. Enciso, I.Fortes, Closure via functional dependence simplification, International Journal of Computer Mathematics, 89(4): 510–526, 2012.
13. G. Stumme, Attribute Exploration with Background Implications and Exceptions. Data Analysis and Information Systems. Statistical and Conceptual approaches. Proc. GfKl'95. Studies in Classification, Data Analysis, and Knowledge Organization 7: 457–469, 1996.
14. L. Szathmary and A. Napoli and S. O. Kuznetsov, ZART: A Multifunctional Itemset Mining Algorithm , Proc. of the 6th Intl. Conf. on Concept Lattices and Their Applications (CLA '08): 47–58, 2008.
15. L. Szathmary and P. Valtchev and A. Napoli and R. Godin, An Efficient Hybrid Algorithm for Mining Frequent Closures and Generators, Proc. of the 5th Intl. Conf. on Concept Lattices and Their Applications (CLA '07): 26–37, 2007.
16. L. Szathmary and P. Valtchev and A. Napoli and R. Godin, Efficient Vertical Mining of Frequent Closures and Generators, LNCS 5772: 393–404, 2009.
17. K. Nehmé, P. Valtchev, M. H. Rouane, R. Godin, On Computing the Minimal Generator Family for Concept Lattices and Icebergs, LNCS 3403: 192–207, 2005.
18. R. Wille, Restructuring lattice theory: an approach based on hierarchies of concepts. In I. Rival (ed.), Ordered sets, pp. 445-470, 1982.