NextPriorityConcept: A new and generic algorithm computing concepts from complex and heterogeneous data

Christophe Demko, Karell Bertet, Cyril Faucher, Jean-François Viaud, Sergei Kuznetsov

Version of Record: https://www.sciencedirect.com/science/article/pii/S0304397520304837
Manuscript_0b8d2289e741f1307349edf030a7d73d

NextPriorityConcept: A new and generic algorithm computing concepts from complex and heterogeneous data

Christophe Demko[a,*], Karell Bertet[a], Cyril Faucher[a], Jean-François Viaud[a], Sergei O. Kuznetsov[b]

[a] *L3i, La Rochelle University, France*

[b] *National Research University Higher School of Economics, Moscow, Russia*

Abstract

In this article, we present a new data type agnostic algorithm calculating a concept lattice from heterogeneous and complex data. Our NextPriorityConcept algorithm is first introduced and proved in the binary case as an extension of Bordat's algorithm with the notion of strategies to select only some predecessors of each concept, avoiding the generation of unreasonably large lattices. The algorithm is then extended to any type of data in a generic way. It is inspired by the pattern structure theory, where data are locally described by predicates independent of their types, allowing the management of heterogeneous data.

Keywords: Formal Concept Analysis, Lattice, Pattern Structures, Strategies, Heterogeneous data

1. Introduction

Formal Concept Analysis (FCA) is a branch of applied lattice theory, which originated from the study of relationship between Galois connections, closure operators, and orders of closed sets [1, 2].

Starting from a binary relation between a set of objects and a set of attributes, formal concepts are built as maximal sets of objects in relation with maximal sets of attributes, by means of derivation operators forming a Galois connection whose composition is a closure operator [3]. Concepts form a partially ordered set, called concept lattice, which represents the initial data. This lattice has proved to be useful in many fields, e.g. artificial intelligence, knowledge management, data-mining, machine learning, etc.

*Corresponding author

Email addresses: christophe.demko@univ-lr.fr (Christophe Demko), karell.bertet@univ-lr.fr (Karell Bertet), cyril.faucher@univ-lr.fr (Cyril Faucher), jean-francois.viaud@univ-lr.fr (Jean-François Viaud), skuznetsov@hse.ru (Sergei O. Kuznetsov)

Many extensions from the original formalism, which was based on binary data, have been studied in order to work with non-binary data, such as numbers, intervals, sequences, trees, and graphs. The formalism of pattern structures [4, 5, 6] extends FCA to deal with non-binary data provided by space description organised as a semi-lattice in order to maintain a Galois connection between objects and their descriptions. Therefore, a pattern lattice represents the data where concepts are composed of objects together with their shared descriptions.

Logical Concept Analysis [7] is a generalization of FCA in which sets of attributes are replaced by logical expressions. The power set of attributes mentionned by the Galois connection is replaced by an arbitrary set of formulas to which are associated a deduction relation (i.e. subsumption), and conjunctive and disjunctive operations, and therefore forms a lattice. LCA is the fundamental base of Abstract Conceptual Navigation [8] whose principle is to navigate in a conceptual space where places are logical concepts connected by navigation links.

Whether space description in pattern structures or conceptual space in LCA, these generalized spaces extends FCA to non binary data. However, they must be defined as a semi-lattice in a preliminary step, independently of the data, often with a large number of generated concepts and unreasonably large lattices that are uneasy to interpret. Pattern lattices are huge, often untractable [**?**], and the need of approaches to drive the search towards the most relevant patterns is a current challenge. Moreover, pattern structures do not allow an easy management of heterogeneous datasets where several kinds of characteristics describe data.

In this paper, we present the NextPriorityConcept algorithm that computes a concept lattice from heterogeneous data, where:

- Patterns are locally selected and discovered:
  Indeed, patterns of each concept are locally discovered, and predecessors of a concept can be filtered according to a specific strategy. So patterns computed by our algorithm are more adapted to the data, and lattices are smaller.
- Pattern mining for heterogeneous and complex data:
  These patterns are formalized by predicates whatever the description of data, then we can merge patterns issued from distinct space descriptions, and manage heterogeneous data in a generic and agnostic way.

2. Preliminaries

2.1. Formal Concept Analysis

Let $\langle G, M, I\rangle$ be a *formal context* where G is a non-empty set of objects, M is a non-empty set of attributes and $I \subseteq G \times M$ is a binary relation between the set of objects and the set of attributes. Let $(2^G, \subseteq) \underset{\alpha}{\overset{\beta}{\leftrightarrows}} (2^M, \subseteq)$ be the corresponding *Galois connection* where:

- $\alpha : 2^G \to 2^M$ is an application which associates a subset $B \subseteq M$ to every subset $A \subseteq G$ such that $\alpha(A) = \{b \ : \ b \in M \wedge \forall a \in A, aIb\}$;

- $\beta : 2^M \rightarrow 2^G$ is an application which associates a subset $A \subseteq G$ to every subset $B \subseteq M$ such that $\beta(B) = \{a \ : \ a \in G \wedge \forall b \in B, aIb\}$.

A concept is a pair (A, B) such that $A \subseteq G$, $B \subseteq M$, $B = \alpha(A)$ and $A = \beta(B)$. The set A is called the *extent*, whereas B is called the *intent* of the concept (A, B). There is a natural hierarchical ordering relation between the concepts of a given context which is called the subconcept-superconcept relation:

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

The ordered set of all concepts makes a complete lattice called the *concept lattice* of the context, that is, every subset of concepts has an infimum (meet) and a supremum (join).

2.2. A basic algorithm

Bordat's theorem [9] states that there is a bijection between the immediate successors of a concept (A, B) and the inclusion maximal subsets of the following family defined on the objects G:

$$\mathcal{FS}_{(A,B)} = \{\alpha(a) \cap B \ : \ a \in G \setminus A\} \tag{1}$$

Bordat's algorithm [9], that we also find in Linding's work [10], is a direct implementation of Bordat's theorem. This algorithm recursively computes the Hasse diagram of the concept lattice of a context $\langle G, M, (\alpha, \beta)\rangle$ starting from the bottom concept $(\beta(M), M)$ and by computing at each recursive call the immediate successors of a concept (A, B): the family $\mathcal{FS}_{(A,B)}$ is first computed, then the inclusion maximal sets are selected.

$\mathcal{FS}_{(A,B)}$ is composed of the intent part of the immediate potential successors of (A, B). Each intent B' is obtained by $\alpha(a) \cap B$, where $a \in G \setminus A$ is a new potential object for a successor concept of (A, B). Indeed, as defined by the order relation between concepts, immediate successors of (A, B) are obtained by an increase of A by at least one new potential object a, and thus a reduction of B to $B' = \alpha(a) \cap B$, and clearly $B' \subset B$. Moreover, B' must be maximal by inclusion in $\mathcal{FS}_{(A,B)}$, otherwise there exists $B'' \in \mathcal{FS}_{(A,B)}$ such that $B' \subset B'' \subset B$, then B' is not the intent of an immediate successor of (A, B). If B' is inclusion maximal in $\mathcal{FS}_{(A,B)}$, then $(\beta(B'), B')$ is an immediate successor of (A, B).

Our NextPriorityconcept algorithm focuses on the objects and the dual version of Bordat's theorem. It considers the whole set G of objects at the beginning. Then, for each concept (A, B), it does not test a new potential object, but a new potential attribute $b \in M \setminus B$ describing a subset of A. In this way, A decreases while B increases, which corresponds to the predecessor relation. This process corresponds to the dual version of Bordat's theorem stating that the immediate predecessors of (A, B) are the maximal inclusion subsets of the following family on the attributes M:

$$\mathcal{FP}_{(A,B)} = \{\beta(b) \cap A \ : \ b \in M \setminus B\} \tag{2}$$

The NextPriorityConcept-Basic algorithm is a new version of Bordat's algorithm where recursion is replaced by a priority queue using the support of concepts. At each iteration, the concept (A, B) of maximal support is produced, then its immediate predecessors are computed by Predecessors-Basic $((A, B))$ that returns the inclusion maximal sets of $\mathcal{FP}_{(A,B)}$, and then they are stored in the priority queue. Therefore, concepts are generated level by level, starting from the top concept $(G, \alpha(G))$, and each concept is generated before its predecessors.

Data:
- $\langle G, M, (\alpha, \beta)\rangle$ be a formal context

Output:
- concepts (A, B) of the formal context

begin
  `/* Priority queue for the concepts */`
  $Q \leftarrow$ `[]` ; `/* Q is a priority queue using the support of concepts */`
  Q.push($(|G|, G)$) ; `/* Add the top concept into the priority queue */`

  while Q **not empty do**
    `/* Compute concept */`
    $A \leftarrow Q$.pop() ; `/* Get the concept with highest support */`
    $B \leftarrow \alpha(A)$; `/* Compute the intent of this concept */`
    produce (A, B) ;

    `/* Update queue */`
    $L \leftarrow$ Predecessors-Basic$((A, B), M, (\alpha, \beta))$;
    forall the $A' \in L$ **do**
      Q.push($(|A'|, A')$) ; `/* Add concept into the priority queue */`
    end
  end
end

Algorithm 1: NextPriorityConcept-Basic

Data:
- (A, B) a concept
- M the set of attributes
- (α, β) the Galois connection

Result:
- L a set of predecessors of (A, B) represented by their extent

begin
  $L \leftarrow \emptyset$;
  forall the $b \in M \setminus B$ **do**
    $A' \leftarrow \beta(b) \cap A$; `/* Extent of a new potential predecessor */`
    $L \leftarrow$ Inclusion-Max(L,A') ; `/* Add` A' `if maximal in` L `*/`
  end
  return L
end

Algorithm 2: Predecessor-Basic

Data:
- L a set of potential predecessors represented by their extent
- A a new potential predecessor

Result:
- inclusion maximal subsets of $L \cup \{A\}$

```
begin
    add ← true;
    forall the A' ∈ L do
        if A ⊂ A' then
            add ← false ;                    /* A is not an immediate predecessor */
            break
        else if A' ⊂ A then
            L.remove(A') ;                   /* Remove A' as a possible predecessor */
        end
    end
    if add then L.add(A);                    /* Add A as a predecessor */
    return L
end
```

Algorithm 3: INCLUSION-MAX

This non-recursive version of Bordat's algorithm preserves its complexity. Therefore, we can state the following result:

Property 1. *Algorithm* NEXTPRIORITYCONCEPT-BASIC *computes the concept lattice of* $\langle G, M, (\alpha, \beta)\rangle$ *in* $O(|G||M|^2 \cdot |\mathcal{B}|)$ *time.*

We will introduce our NEXTPRIORITYCONCEPT algorithm in two steps:

In Section 3, NEXTPRIORITYCONCEPT-BASIC algorithm is first modified in order to introduce the possibility to filter the new attributes considered during the immediate predecessor process according to a *strategy* σ of exploration.

In Section 4, the final version of NEXTPRIORITYCONCEPT, inspired by pattern structures, extends the computation of concepts to heterogeneous dataset, where attributes P are predicates deduced from each characteristics of data according to a specific *description*.

3. NextPriorityConcept: filtering of concepts according to a strategy

3.1. Extension of the algorithm with strategies

For real data lattices are often too large, which hinders their ability to provide readability and explanation of the data. In this section, we extend the basic NEXTPRIORITYCONCEPT-BASIC algorithm to select only some predecessors at each iteration. Rather than considering all the attributes of $M \setminus B$ to calculate the potential predecessor of a concept (A, B), we apply a filter on these candidate attributes. For example, we can select attributes of maximal support, or according to class information as explained later.

More formally, a *strategy* is an input application $\sigma : 2^G \to 2^M$ which associates a subset $S \subseteq M$ of selected attributes to every subset $A \subseteq G$. Many strategies are possible. Let

us introduce as examples the maximal support strategy $\sigma_{\max}$ and the entropy strategy σ_{entropy}:

- The maximal support strategy relies on the support of attributes:

$$\sigma_{\max}(A) = \{b \in M \setminus \alpha(A) \ : \ |\beta(\alpha(A) \cup \{b\})| \text{ maximal}\}$$

- The entropy strategy is a supervised strategy where objects have a *class* attribute:

$$\sigma_{\text{entropy}}(A) = \{b \in M \setminus \alpha(A) \ : \ H_{class}(\beta(\alpha(A) \cup \{b\})) \text{ minimal}\}$$

We introduce a new set P of selected attributes according to the strategy, and to avoid confusion, we will denote (A, D) a concept defined on $G \times P$, and I_P the corresponding relation between G and P. An attribute $p = \bigwedge(D \cup \{b\})$ is added in P when $b \in M$ is selected by the strategy σ, i.e. when $b \in \sigma(A)$. Then b is in relation in the final context only with some objects in A, meaning "b only for the objects sharing D", denoted $b|D$. We can observe that the same attribute $b \in M$ can be selected several times with distinct meanings, $b|D$ and $b|D'$ are not the same when $D \neq D'$ and produce distinct attributes in P. Therefore, P is not included in M.

To ensure that meets are correctly generated, we introduce a constraints propagation mechanism $\mathcal{C}$ that associates a set of attributes $\mathcal{C}[A]$ to process with each concept (A, D). More formaly, this propagation mechanism is an application $\mathcal{C}$ defined for the intent of any concept (A, D) to 2^P by $\mathcal{C}[A] = C_{residual} \cup C_{cross}$ where

- $C_{residual}$ is the set of residual constraints issued from the successor concept (A', D') that generated (A, D) :
$$C_{residual} = \mathcal{C}[A'] \backslash D$$

- C_{cross} is the set of cross constraints issued from the others predecessors (A_i, D_i) of (A, D):
$$C_{cross} = (\bigcup_i A_i \cap \sigma(A')) \backslash D$$

- and $\mathcal{C}[G] = \emptyset$

The Predecessors-Strategy algorithm is a modified version of Predecessors-Basic to compute predecessors of a concept (A, D), where we only consider the attributes $\sigma(A)$ given by the strategy, instead of the whole set $M \setminus D$ of attributes. To ensure that meets will be computed, we also consider attributes issued from neigbor concepts through the constraint propagation mechanism $\mathcal{C}[\mathcal{A}]$.

The NextPriorityConcept-Strategy algorithm considers a formal context $\langle G, M, (\alpha, \beta)\rangle$ and a strategy σ as input, and computes the concept lattice of $\langle G, P, I_P\rangle$ according to the input strategy in the same way of the NextPriorityConcept-Basic algorithm, with the additional control of the constraint propagation.

Inclusion-Max is similar, with a minimal test on the subset of objects A', but the list L is composed of pairs (A', d) instead of subsets A'.

3.2. Proof and complexity analysis

3.2.1. Proof of the algorithm

Theorem 1. *The* NextPriorityConcept-Strategy *algorithm computes all the concepts of* $\langle G, P, (\alpha_P, \beta_P)\rangle$*, with a strategy* σ *as input.*

Consider (A_i, D_i) the concept generated at each iteration i of the main loop. To prove the theorem, we have to prove the two following lemmas:

Lemma 1. *Let* (A_i, D_i) *be a concept generated at iteration* i*, then* (A_i, D_i) *is a concept of* $\langle G, P, (\alpha_P, \beta_P)\rangle$

Proof

The priority queue Q is initialized with $(|G|, (G, \alpha \circ \beta(\emptyset)))$, and $(G, \alpha \circ \beta(\emptyset)) = (A_0, D_0)$ corresponds to the top concept on P, i.e. the concept of greatest support.

Let us introduce P_i the set of selected attributes P at each iteration i. Since P is updated with new selected attributes at each iteration i, we have $P_0 \subseteq P_1 \subseteq \ldots \subseteq P_i \ldots \subseteq P$ with P_0 being initialized with $D_0 = \alpha \circ \beta(\emptyset)$.

Let (A_i, D_i) be the concept generated at iteration i. Let us prove that (A_i, D_i) is a concept of the generated context $\langle G, P, I_P\rangle$.

Clearly $D_i \subseteq P_i$ and, since $(A_i \times D_i)$ is added in I_P at iteration i, then (A_i, D_i) is a concept of the context $\langle G, P_i, I_P\rangle$. Therefore we have to prove that (A_i, D_i) is also a concept of $\langle G, P, I_P\rangle$.

If (A_i, D_i) is not a concept of $\langle G, P, I_P\rangle$, then there exists a selector $p \in P$ such that $p \in \alpha(A_i)$ and $p \notin D_i$, thus $p \notin P_i$. From $p \in \alpha(A_i)$ we have $A_i \subseteq \beta(p)$. Let the iteration j that adds p in P, and let (A_j, D_j) the concept generated at iteration j. Then $p \in D_j \subseteq P_j$ and $(A_j \times D_j)$ is added in I_P, thus $A_j = \beta(p)$. From $p \notin P_i$ and $p \in P_j$, we deduce $j > i$. From $A_i \subseteq \beta(p)$ and $A_j = \beta(p)$, we deduce $A_i \subset A_j$ and $j < i$ since concept are generated according to the priority queue using the support. Thus a contradiction and (A_i, D_i) is a concept of $\langle G, P, I_P\rangle$.

□

Lemma 2. *Let* (A, D) *be a concept of* $\langle G, P, (\alpha_P, \beta_P)\rangle$*, then there exists an iteration* i *such that* $(A, D) = (A_i, D_i)$*.*

Proof

Let (A, D) be a concept of $\langle G, P, I_P\rangle$. Then (A, D) is the meet of $\{(\beta(p), \alpha \circ \beta(p))_{p \in D}\}$. Let us prove that this meet is generated by the algorithm.

In the case where $|D| = 0$, then (A, D) is the top concept generated at the begining of the algorithm. In the case where $|D| = 1$, then $(A, D) = (\beta(p), \alpha \circ \beta(p))$ generated by the iteration i that adds p in P.

In the case where $|D| > 1$, let $p \neq p' \in D$ such that $i < j$, where i is the iteration that adds p in P, and j is the iteration that adds p' in P.

Let (A_i, D_i) be the concept generated at iteration i and (A_j, D_j) be the concept generated at iteration j. Then $(A_i, D_i) = (\beta(p), \alpha \circ \beta(p))$ and $(A_j, D_j) = (\beta(p'), \alpha \circ \beta(p'))$.

Let us prove that the meet $(A_i, D_i) \wedge (A_j, D_j)$ is generated. We have two cases:

- If $D_i \subset D_j$ then $(A_i, D_i) \leq (A_j, D_j)$ and (A_j, D_j) is equal to $(A_i, D_i) \wedge (A_j, D_j)$.
- If $D_i \not\subset D_j$ then the iteration i adds p as constraint to all other concepts, thus p belongs to the set of constraints of A_j, and is considered is the first loop of the Predecessors-Strategy algorithm to generate a potential immediate predecessor $S = (\beta(D_j \cup \{p\}), \alpha \circ \beta(D_j \cup \{p\}))$ of (A_j, D_j). We have two cases again.
  - In the case where S is inclusion minimal among all the potential immediate predecessors of (A_j, D_j), then S is generated as immediate predecessor and corresponds to the meet $(A_i, D_i) \wedge (A_j, D_j)$.
  - In the case where S is not inclusion minimal, then p belongs to the set of constraints of the generated immediate predecessors of (A_j, D_j), and the meet $(A_i, D_i) \wedge (A_j, D_j)$ will be generated as a predecessor of an immediate predecessor of (A_j, D_j).

Therefore, the meet $(A_i, D_i) \wedge (A_j, D_j)$ is generated thanks to the constraints propagation mechanism, which completes the proof.

□

Hence, we can state the following result considering the selectors $p \in \sigma(A) \cup \mathcal{C}[A]$:

Lemma 3. *There is a bijection between the immediate predecessors of a concept (A, D) and the inclusion maximal subsets of the following family defined on the objects G:*

$$\mathcal{FD}_{(A,D)} = \{\{a \in A \ : \ p(a)\} : \ p \in (\sigma(A) \cup \mathcal{C}[A])\} \tag{3}$$

The context $\langle G, P, I_P \rangle$ generated by the strategy uses attributes defined in M. Each attribute p of the context $\langle G, P, I_P \rangle$ is constructed as a conjunction of a non-empty subset of M. This is obviously not a subcontext of the context $\langle G, M, (\alpha, \beta) \rangle$ but each attribute of the context $\langle G, P, I_P \rangle$ is subsumed by at least an attribute of the context $\langle G, M, (\alpha, \beta) \rangle$.

3.2.2. Run-time analysis

Denote by $\mathcal{B}$ the collection of all formal concepts of $\langle G, M, (\alpha, \beta) \rangle$ generated by the strategy σ, i.e. the concepts of $\langle G, P, (\alpha_P, \beta_P) \rangle$; and by c_σ the cost of the strategy for a concept.

- the Predecessors-Strategy algorithm computes the predecessors of a concept. Its run-time complexity is $O(|G|\,|P|^2\,c_\sigma)$ where $O(|G|\,|P|^2)$ is the cost of Bordat's algorithm [9]. And the descendant constraints are updated in $O(|P|^2)$.
- the NextPriorityConcept-Strategy algorithm updates the priority queue in $O(|G|\,|P|)$.

Therefore we can deduce the run-time complexity of the NextPriorityConcept-Strategy algorithm: $O(|\mathcal{B}|\,|G|\,|P|^2\,c_\sigma)$.

3.2.3. Memory analysis

At each step of the main loop in the NextPriorityConcept-Strategy algorithm, a set of predecessors is generated. The cardinality of these predecessors cannot exceed $|P|$. These predecessors will not be explored until all their predecessors have been examined (principle of the priority queue using the support of concepts). For each concept in the priority queue, a set of constraints is maintained whose cardinality cannot exceed $|P|$. So the memory complexity of the NextPriorityConcept-Strategy algorithm is in $O(w\,|P|^2)$ where w is the width of the concept lattice.

3.3. Example

Table 1: `Digit` context where **c** stands for **c**omposed, **e** for **e**ven, **o** for **o**dd, **p** for **p**rime and **s** for **s**quare

	c	**e**	**o**	**p**	**s**
0	✓	✓			✓
1			✓		✓
2		✓		✓	
3			✓	✓	
4	✓	✓			✓
5			✓	✓	
6	✓	✓			
7			✓	✓	
8	✓	✓			
9	✓		✓		✓

Table 2: Execution

Step	$\mathcal{C}$	Q	(A, P)	$\lvert A\rvert$
0	{0123456789:∅}	[(10,$0)]	(0123456789,∅)	10
1	{04689:eo, 02468:co, 13579:ce}	[(5,$1), (5,$2), (5,$3)]	(04689,c)	5
2	{02468:co, 13579:ce, 0468:o}	[(5,$2), (5,$3), (4,$4)]	(02468,e)	5
3	{13579:ce, 0468:o}	[(5,$3), (4,$4)]	(13579,o)	5
4	{0468:o, 357:e, 9:e}	[(4,$4), (3,$5), (1,$7)]	(0468,ce)	4
5	{357:e, 9:e, 04:o}	[(3,$5), (2,$6), (1,$7)]	(357,op)	3
6	{9:e, 04:o, ∅:ceops}	[(2,$6), (1,$7), (0,$8)]	(04,ces)	2
7	{9:e, ∅:ceops}	[(1,$7), (0,$8)]	(9,cos)	1
8	{∅:ceops}	[(0,$8)]	(∅,ceops)	0
9	{}		[]	

Table 3: Context of the lattice with the maximal support strategy where (✓) stands for the relations that are not considered

	c	**e**	**o**	**p|o**	**s|ec**
0	✓	✓			✓
1			✓		(✓)
2		✓		(✓)	
3			✓	✓	
4	✓	✓			✓
5			✓	✓	
6	✓	✓			
7			✓	✓	
8	✓	✓			
9	✓		✓		(✓)

We consider the formal context `digit` in Table 1 as the first example. The classical concept lattice produced without strategy is displayed in Figure 2. The maximal support strategy $\sigma_{\max}$ leads to the lattice whose Hasse diagram is displayed in Figure 1, where attributes and objects are indicated in respectively the first and the last concept where they appears. We also indicate the number (using \$) and the support (using #) of each concept. The trace execution is in Table 2. The resulting context (G, P, I_P) displayed in Table 3 has been constructed using the initial one.

We can observe that this second concept lattice contains only 9 concepts instead of 14. The concepts for attributes **p** and **s** are not generated as immediate predecessors of the top concept since their support is not maximal. However, **p** appears in concept \$5, generated as a predecessor of concept \$3 equal to $(\{1, 3, 5, 7, 9\}, o)$, thus introduced only for the `odd` digits $\{1, 3, 5, 7, 9\}$.

Therefore, **p** means `prime` property only for the `odd` digits, denoted by **p|o**. In the same way, **s** appears in concept \$6 as a predecessor of concept \$4 equal to $(\{0, 4, 6, 8\}, ec)$, meaning `square` property for the `even` and `composite` digits, denoted by **p|ec**.

As second example, we consider the `Lenses` dataset from the UCI Machine Learning Repository[1]. This dataset is composed of 24 objects/patients described by 4 categorical attributes:

- *age of the patient*: young, pre-presbyopic, presbyopic
- *spectacle prescription*: myope, hypermetrope
- *astigmatic*: no, yes
- *tear production rate*: reduced, normal

and classified in 3 classes

- the patient should be fitted with *hard contact lenses*,
- the patient should be fitted with *soft contact lenses*,

[1] https://archive.ics.uci.edu

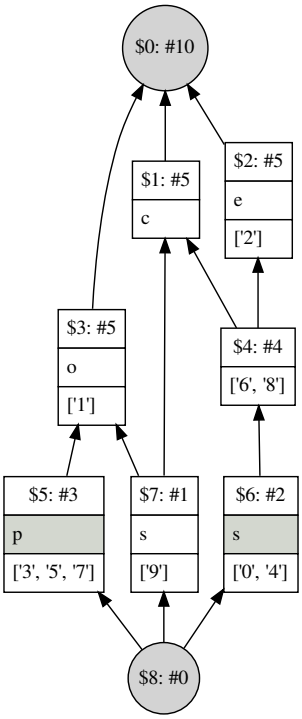

Figure 1: `Digit` sample with greatest support strategy

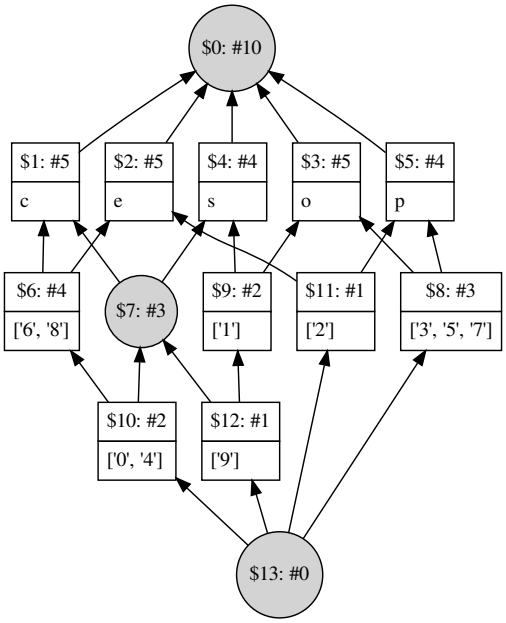

Figure 2: `Digit` sample without strategy

- the patient should *not be fitted* with contact lenses.

We consider the formal context composed of 9 binary attributes, i.e. the modalities of the 4 categorical attributes.

The classical concept lattice contains 109 concepts. With the entropy strategy σ_{entropy} using the class information and by keeping only the two best entropy measures for the predecessors, we obtain a more compact lattice of 28 concepts displayed in Figure 3.

As long as a concept contains only one class, no new predecessors are generated by the strategy. Therefore, these concepts and their upper neighbors can be interpreted as a clustering of the data, each concept (A, D) among these clusters corresponding to a class c, and meaning that objects having attributes D belong to the class c.

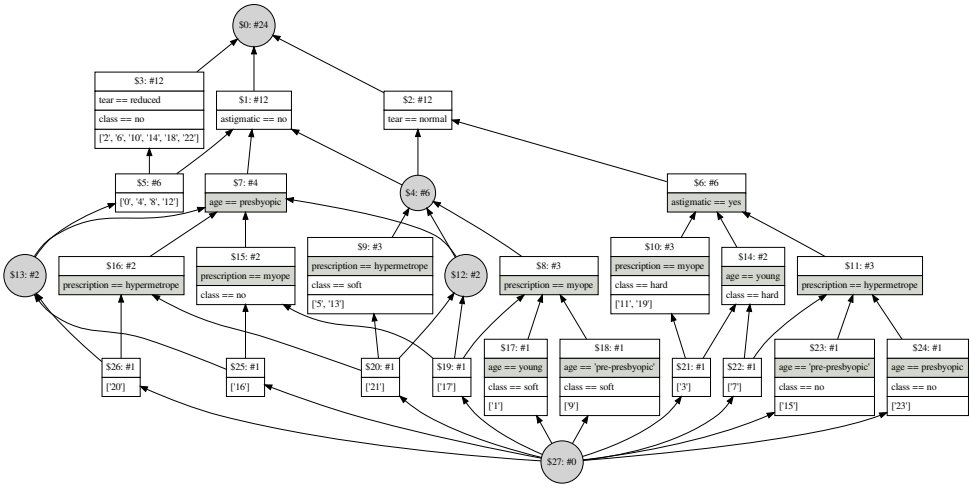

Figure 3: `Lenses` dataset with the entropy strategy

4. NextPriorityConcept: heterogeneous data as input

4.1. NextPriorityConcept *algorithm: the final version*

In the previous section, the new set P of attributes is introduced to store the selected attributes from which predecessors of a concept (A, D) are generated. It is easily observed that it is also possible to introduce new attributes at each iteration without changes. For example, we can consider $\sigma_{\text{neg}}(A) = \{b, \overline{b} \ : \ b \in M - \alpha(A)\}$, a strategy adding negative attributes as selector.

Our final NextPriorityConcept algorithm exploits this possibility to manage heterogeneous data as input. We use predicates describing objects independent of their types. In order to avoid confusion with classical binary attributes, we will use "characteristic" instead of "attribute".

More formally, we consider a heterogeneous dataset (G, S) as input where each **characteristic** $s \in S$ can be seen as a mapping $s : G \to R_s$ where R_s is called the domain of s. Let p_s be a **predicate** for a characteristic s and $a \in G$, we write $p_s(a)$ when $s(a)$ verifies p_s.

For example, a numerical characteristic can be described by predicates of the form *is smaller/greater than c* where c a numerical value; a characteristic representing a (temporal) sequence can be described by predicates of the form *contains s as (maximal) subsequence* where s is a sequence; and a classical boolean characteristic b corresponds to the predicate *possesses b*.

We introduce the notion of *description* δ to provide predicates describing a set of objects, and we extend the notion of strategy σ to provide predicates (called selectors) to generate (select) the predecessors of a concept.

Characteristics of different domains must be processed separately since predicates are calculated differently according to the domain. However, characteristics of the same domain can be processed together or separately, and some characteristics may not be considered, or considered several times.
Therefore, characteristics are given by a family $S = (S^i)_{i \leq d}$, where each S^i contains characteristics of the same domain.

For example, for the well-known Iris database[2] composed of class information and four numerical characteristics $S = \{\texttt{sepal-length}, \texttt{sepal-width}, \texttt{petal-length}, \texttt{petal-width}\}$, we can consider the `petal` characteristics together, and the `sepal` characteristics together ($S^1 = \{\texttt{petal-length}, \texttt{petal-width}\}$ and $S^2 = \{\texttt{sepal-length}, \texttt{setal-width}\}$). We can also only consider the petal characteristics, and separately ($S^1 = \{\texttt{petal-length}\}$ and $S^2 = \{\texttt{petal-width}\}$).

Predicates are provided according to a given S^i, both to describe a set of objects, but also to compute the predecessors of a concept:

A description δ^i is an application $\delta^i : 2^G \to 2^P$ which defines a set of predicates $\delta^i(A)$ describing the characteristics of S^i for any subset A of G.

A strategy σ^i is an application $\sigma^i : 2^G \to 2^P$ which defines a set of predicates $\sigma^i(A)$ (called selectors) for characteristics of S^i from which the predecessors of a concept (A, D) are generated.

Therefore, the strategy $\sigma(A)$ and the description $\delta(A)$ of a subset A of objects are defined from 2^G to 2^P by:

$$\delta(A) = \bigcup_{S^i} \delta^i(A)$$

$$\sigma(A) = \bigcup_{S^i} \sigma^i(A)$$

Let us give some examples of strategies and descriptions for a set $A \subseteq G$ of objects:

[2] https://archive.ics.uci.edu

- For a numerical attribute $S^i = \{s\}$:
  - $\delta^i(A) = \{$*is greater than* $\min_{a\in A} s(a)$, *is smaller than* $\max_{a\in A} s(a)\}$
  - $\sigma^i(A) = \{$*is greater than* q_1, *is smaller than* $q_3\}$ where q_1 and q_3 are respectively the first and the third quantile of the values $(s(a))_{a\in A}$.
- For a sequential attribute $S^i = \{s\}$ [11]:
  - $\delta^i(A) = \{$ *contains* X *as subsequences*$\}$ where X is the set of longest most common subsequences of $\{s(a)\}_{a\in A}$.
  - $\sigma^i(A) = \{$ *contains* X'*as subsequences*$\}$ where sequences of X' are obtained by an augmentation process of the longest most common subsequences of $\delta^i(A)$, and thus a reduction of the sequences A' containing these subsequences.

NextPriorityConcept and Predecessors-Description algorithms are very similar to the previous versions. NextPriorityConcept algorithm considers as input

- a heterogeneous dataset (G, S)
- $(S^i)_{i\leq d}$ a family of S
- δ a description
- σ a strategy

This algorithm computes the formal context $\langle G, P, I_P\rangle$ and its concepts, where P is the set of predicates describing the characteristics, $I_P = \{(a, p) \ : \ p(a)\}$ is the relation between objects and predicates, and (α_P, β_P) is the associated Galois connection.

Predecessors-Description is a modified version of Predecessors-Strategy to compute predecessors of a concept (A, D), where we consider $\sigma(A)$ and $\delta(A)$ given as input.

4.2. Discussions

4.2.1. Comparison with pattern structures

The predicates in the final set P are those issued from the descriptions since the predicates generated by the strategy are only used to generate predecessors. Our NextPriority-Concept algorithm can be run on a pattern structure over each domain of characteristics S^i.

Formally, a pattern structure [4] is a triple $(G, (\mathcal{D}, \sqcap), \delta)$ where G is a set of objects, $(\mathcal{D}, \sqcap)$ is a meet semi-lattice of potential objects descriptions, and $\delta : G \to \mathcal{D}$ associates to each object its description. Elements of $\mathcal{D}$ are ordered by the subsumption relation $\sqsubseteq$.

Let $(2^G, \subseteq) \underset{\alpha_\mathcal{D}}{\overset{\beta_\mathcal{D}}{\leftrightarrows}} (\mathcal{D}, \sqcap)$ be the corresponding Galois connection where:

- $\alpha_\mathcal{D} : 2^G \to \mathcal{D}$ is defined for $A \subseteq G$ by $\alpha_\mathcal{D}(A) = \sqcap_{g\in A}\delta(g)$.
- $\beta_\mathcal{D} : \mathcal{D} \to 2^G$ is defined, for $d \in \mathcal{D}$ by $\beta_\mathcal{D}(d) = \{g \in G \ : \ d \sqsubseteq \delta_\mathcal{D}(d)\}$.

Pattern concepts are pairs (A, d), $A \subseteq G$, $d \in \mathcal{D}$ such that $\alpha_\mathcal{D}(A) = d$ and $A = \beta_\mathcal{D}(d)$. d is a pattern intent, and is the common description of all objects in A. When partially ordered by $(A_1, d_1) \leq (A_2, d_2) \iff A_1 \subseteq A_2 (\iff d_2 \sqsubseteq d_1)$, the set of all pattern concepts forms a lattice called the *pattern lattice*.

Theorem 2. *If each description δ^i verifies $\delta^i(A) \sqsubseteq \delta^i(A')$ for $A' \subseteq A$, then* NextPriorityConcept *algorithm computes the concept lattice of $\langle G, P, (\alpha_P, \beta_P)\rangle$ with $O(|\mathcal{B}|\,|G|\,|P|^2\,(c_\sigma + c_\delta))$ run-time (where $\mathcal{B}$ is the number of concepts, and c_σ and c_δ are the costs of the strategy and the description depending on the algorithms chosen to describe the data and to select the predecessors), and using $O(w\,|P|^2)$ space memory (where w is the width of the concept lattice).*

Proof

We have an implicit description space $\mathcal{D}^i$ by δ^i for each subset S^i of characteristics, where the description $\delta^i(A)$ of a set $A \subseteq G$ is a direct translation by predicates of its description in $\mathcal{D}^i$. A nice result in pattern structure establishes that there is a Galois connection between G and $\mathcal{D}^i$ if and only if $(\mathcal{D}^i, \sqcap)$ is a complete meet semi-lattice. Therefore, in order to maintain the final Galois connection (α_P, β_P) between objects and predicates, each description $\delta^i(A)$ must verify $\delta^i(A) \sqsubseteq \delta^i(A')$ for $A' \subseteq A$ since $\delta^i(A) \sqsubseteq \delta^i(A') \iff \delta^i(A) \sqcap \delta^i(A') = A$.

Therefore, since NextPriorityConcept algorithm is a direct extension of NextPriorityConcept-Strategy algorithm to manage the set P of predicates issued from descriptions δ^i, since the Galois connection is maintained between G and P, and since NextPriorityConcept-Strategy algorithm computes all the concepts of $\langle G, P, (\alpha_P, \beta_P)\rangle$, we can deduce that NextPriorityConcept algorithm computes all the concepts of $\langle G, P, (\alpha_P, \beta_P)\rangle$ with the same complexity.

□

Corollary 1. *If each description δ^i verifies $\delta^i(A) \sqsubseteq \delta^i(A')$ for $A' \subseteq A$, then*

$$(2^G, \subseteq) \underset{\alpha_P}{\overset{\beta_P}{\leftrightarrows}} (2^P, \sqsubseteq) \text{ is a Galois connection}$$

and $\alpha_P \circ \beta_P$ is a closure operator on P.

Proof This corollary is a direct consequence of Theorem 2. Since NextPriorityConcept algorithm computes the concept lattice of $\langle G, P, (\alpha_P, \beta_P)\rangle$, then (α_P, β_P) is a Galois connexion between 2^G and 2^P. Then, because of this Galois connexion, the composition $\alpha_P \circ \beta_P$ is a closure operator. Moreover, we can observe that α_P corresponds to the description δ that associates a set of predicates to any subset $A \subseteq G$. □

While patterns are globally computed in a preprocessing step using pattern structures, our NextPriorityConcept algorithm is a pattern discovery approach where predicates are discovered "on the fly", in a local way for each concept. This is made possible by the use of the priority queue (to ensure that each concept is generated before its predecessors) and the propagation of constraints (to ensure that meet will be computed). Therefore, predicates are well-suited to the data, and lattices are often smaller, with more relevant concepts. Moreover, the use of predicates mixed with specialized strategies and descriptions on each domain of characteristics allows mining of complex and heterogeneous data.

4.2.2. Processing of group of characteristics

When some characteristics are defined on the same domain, the family $(S^i)_{i\leq d}$ offers the possibility to process them separately or together. An immediate way to process with several characteristics together would be to merge the predicates obtained in the individual case, both for the descriptions and for the strategies. But it is possible to obtain more relevant predicates by a specific process of a group of characteristics.

For example, for a group of k numerical characteristics $s_1, \dots s_k$, we can consider the k-dimensional points $\{(s_j(a))_{j\leq k} \ : \ a \in A\}$ for a set A of objects, and their convex hull [12]. The description $\delta^i(A)$ is then composed of predicates describing the borders of the convex hull, and the strategy $\sigma^i(A)$ is a way to cut the hull. For points in two dimensions, the convex hull is a polygon, and borders and cuts are lines. Clearly, for two sets A and A' of objects such that $A' \subseteq A$, the convex hull of A' is included into the convex hull of A, and the intersection of two convex hulls is a convex hull. Therefore $\delta^i(A) \sqsubseteq \delta^i(A')$

For points in two and three dimensions, output-sensitive algorithms are known to compute the convex hull in time $O(n \log n)$, where n is the number of points. For dimensions d higher than 3, the time for computing the convex hull is $O(n^{\lfloor d/2 \rfloor})$ [13]. This process therefore impacts on the costs c_σ and c_δ.

Now consider a group of k boolean characteristics $S^i = \{x_1, \dots, x_k\}$. The classical FCA approach describes a set of objects A by the set of attributes $B = \{x_j \ : \ a \in A \text{ and } x_j(a) = 1\}$ and the strategy of generation of immediate predecessors considers the set of all other attributes $\{x \in S^i \setminus B\}$ as selectors. These two sets described by predicates of the form *possesses attribute* x would respectively correspond to $\delta^i(A)$ and $\sigma^i(A)$.

The use of predicates, and especially the possibility of introducing negative attributes, allows us to consider other descriptions of A. For example, we can consider a description $\delta^i(A)$ by predicates for the disjunction of clauses:

$$\bigvee_{a\in A} \bigwedge_{j\leq k} \begin{cases} x_j & \text{if } x_j(a) = 1 \\ \overline{x_j} & \text{if } x_j(a) = 0 \end{cases}$$

For a finer and minimal description, we can also consider the minimization of this boolean formulae using the well-known Quine-McCluskey algorithm (or the method of prime implicants), with a time complexity in $O(3^n \log n)$ [14] where n is the number of attributes.

4.2.3. About strategies

A strategy proposes a way to *cut* the description $\delta^i(A)$ by selectors from which predecessors of a concept (A, P) are generated. These selected predicates are only used in this way at each step of the algorithm, but are not kept in the final set P of predicates, and several strategies are possible to generate predecessors of a concept (A, P).

Therefore, our algorithm can be extended to improve the strategy management:

Meta-strategy: The strategy σ is defined as the union of the strategies $(\sigma^i)_{i\leq d}$ for each part S^i of attributes. It is possible to introduce a filter (or meta-strategy) on these selectors, as those introduced in Section 3:

- The maximal support meta-strategy relies on the support:

$$\sigma_{\max}(A) = \{p \in \bigcup_{S^i} \delta^i(A) \ : \ |\pi(p)| \text{ maximal}\}$$

- The entropy meta-strategy is a supervised strategy where objects have a class attribute:

$$\sigma_{\text{entropy}}(A) = \{p \in \bigcup_{S^i} \delta^i(A) \ : \ H_{\text{class}}(\pi(p)) \text{ minimal}\}$$

Interactivity: Several strategies are possible to generate predecessors of a concept, going from the naive strategy σ^i_{naive} that generates all the possible predecessors, to the silly strategy $\sigma^i_{\text{silly}} = \emptyset$ that generates no predecessors. Therefore we can extend our algorithm in an interactive way, where the user could choose or test several strategies for each concept in a user driven pattern discovery approach.

In classical FCA approach, the naive strategy considers all the possible attributes of $M \setminus B$ for a concept (A, B), and the corresponding lattice is often too large. The silly strategy allows one to introduce some attributes in concepts, but without considering them in the predecessor generation. This approach is interesting, e.g., for class attributes. Every possible strategy is between σ^i_{naive} and σ^i_{silly} when considering the set of generated predecessors, and it would be interesting to investigate the whole set of possible strategies.
A strategy close to σ^i_{naive} increases the number of concepts, while a strategy close to σ^i_{silly} decreases the number of concepts.

4.2.4. Examples

4.2.4.1. `Iris` *dataset with heterogeneous descriptions.* Consider the well-known `Iris` dataset from the UCI Machine Learning Repository[3], composed of 150 objects described by 4 numerical characteristics `sepal-length`, `sepal-width`, `petal-length`, `petal-width` and classified in 3 classes `Setosa`, `Versicolor`, `Virginica`.

In this example, we consider the two `petal` characteristics separately, and the `class` characteristic, thus a combination of two numerical characteristics with a categorical one. For each `petal` characteristics, we use a classical description by the two predicates *the values are greater than the min* and *the values are smaller than the max*. For the `class` characteristic, we use the predicate *belongs to class*.

[3] https://archive.ics.uci.edu

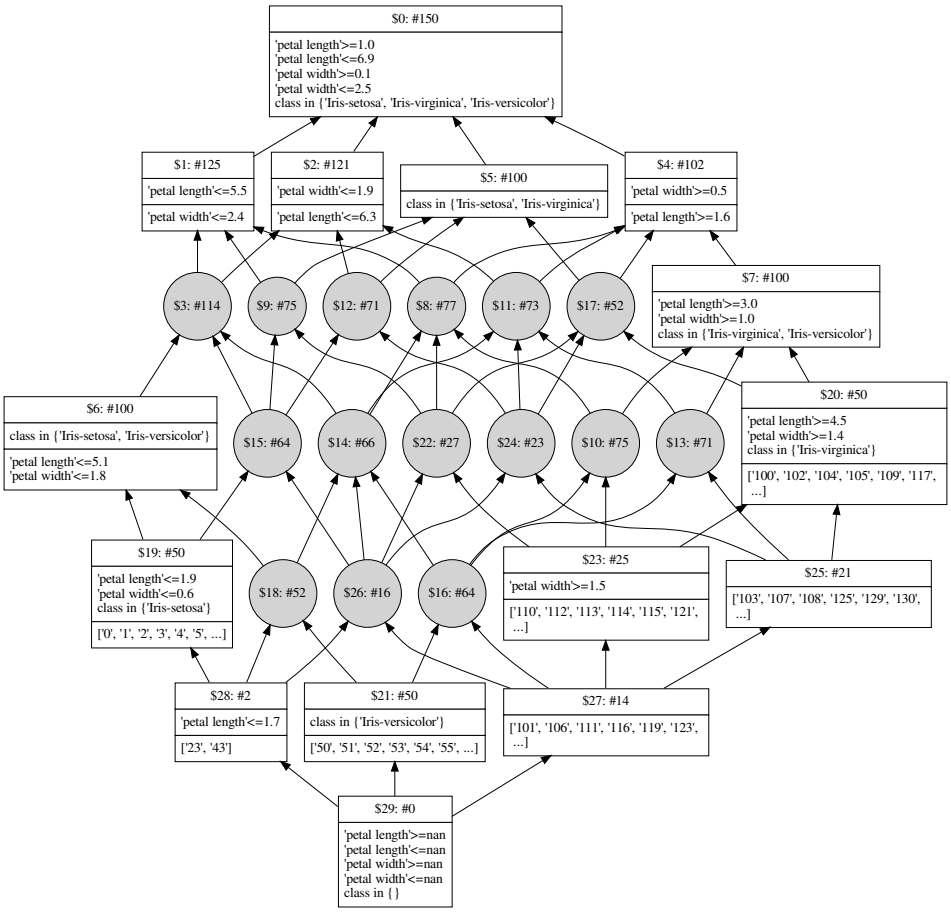

Figure 4: `Iris` dataset with a minimum support (100) strategy using `petal lengh`, `petal width` and `class`

The strategy generates the two selectors *is the value greater than the mean minus the standard deviation?* and *is the value smaller than the mean plus the standard deviation?*, and is combined with a meta-strategy limiting new predecessors to those whose support is greater than 100.

We obtain the concept lattice displayed in Figure 4 composed of 30 concepts:

- concept \$19 corresponds to objects whose class is `Setosa`
- concept \$20 corresponds to objects whose class is `Virginica`
- concept \$21 corresponds to objects whose class is `Versicolor`

4.2.4.2. `Iris` dataset with the entropy strategy. In the second example, we consider the `Iris` dataset and the four `petal` and `sepal` characteristics separately. We use the following entropy strategy which allows one to consider the entropy of a predecessor A' of A, but also the entropy of the remaining set $A \setminus A'$:

$$H = \theta(H_{A'}) + (1 - \theta)H_{A-A'}$$

The more the value of θ increases, the more the number of predessors of A decreases.

By keeping only the predecessors with the best entropy, we obtain the concept lattice displayed in Figure 5 (with $\theta = 1/2$) composed of 13 concepts. The `Setosa` iris are quickly separated according to their two `petal` characteristics (concept \$3). Indeed, we can observe on the scatterplot in Figure 6 that this class is clearly separated from the two others. We obtain 4 concepts for classes `virginica` and `versicolor`:

- concept \$10 and \$11 correspond to objects whose class is `Virginica`, with only the two `petal` characteristics used in concept \$10, while the `sepal-length` characteristic is introduced in concept \$11.
- concept \$5 and \$9 correspond to objects whose class is `Versicolor`,

Finally, we have varied the number k (Figure 7) of best entropy measures kept from 1 to 10 and computed:

- the number of concepts generated;
- the number of meet-irreducible;
- the number of join-irreducible.

Note that the total number of concepts generated would have been equal to 6516292 (Figure 8) if the naive strategy had been chosen (equivalent to keeping all predecessors).

4.2.4.3. Numbers $[36, 48, 56, 64, 84]$ *and all their possible GCD and LCM as descriptions.* In this example, we consider the numbers $[36, 48, 56, 64, 84]$ and the *greatest common divisor (GCD)* and *least common multiple (LCM)* of all their possible subsets as descriptions. The strategy consists in adding a new *is divisor of* or *is multiple of* predicate using a combination of the prime numbers of the set of objects present in the concept.

[4]https://commons.wikimedia.org/wiki/File:Iris_dataset_scatterplot.svg

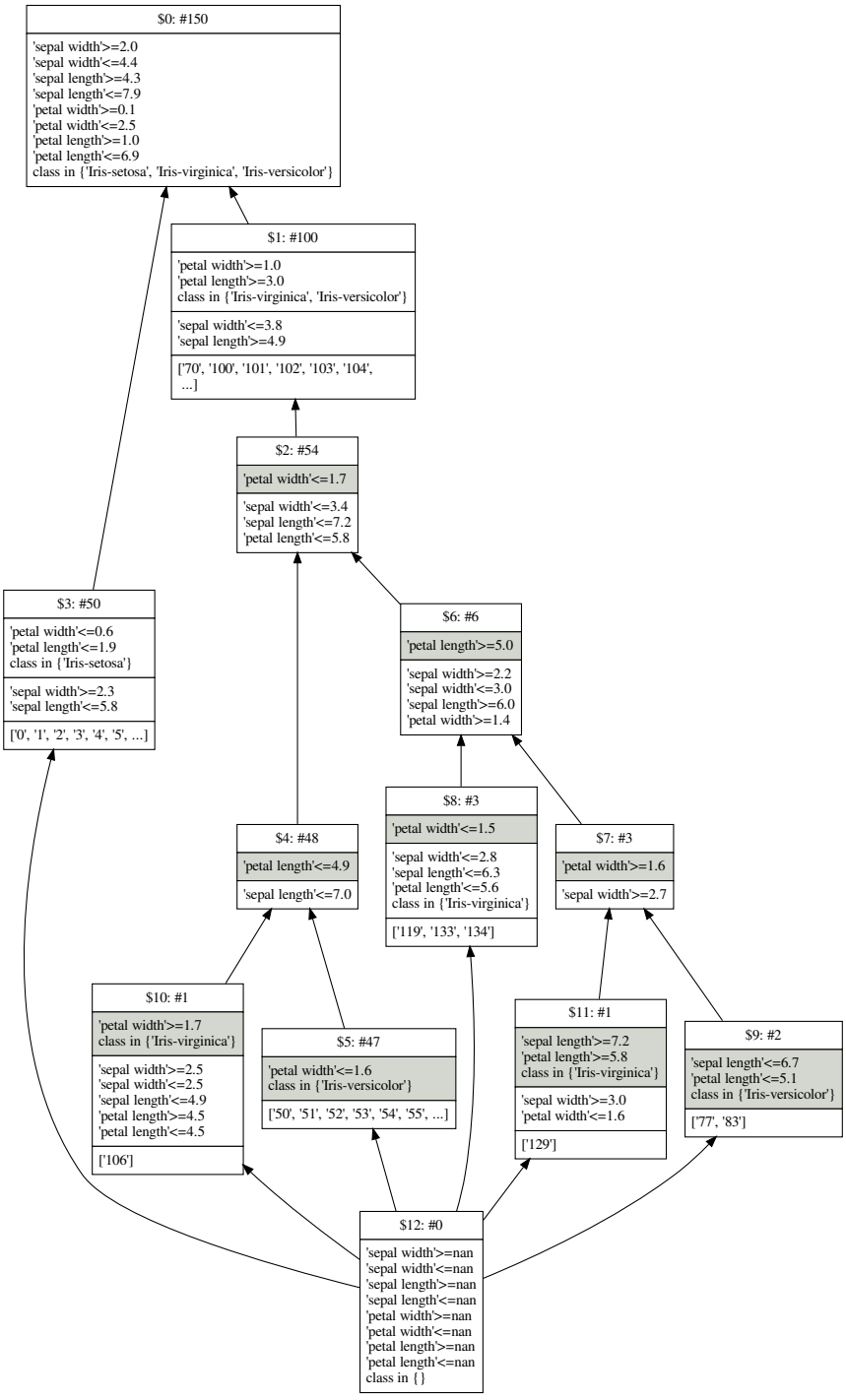

Figure 5: `Iris` dataset with the entropy strategy

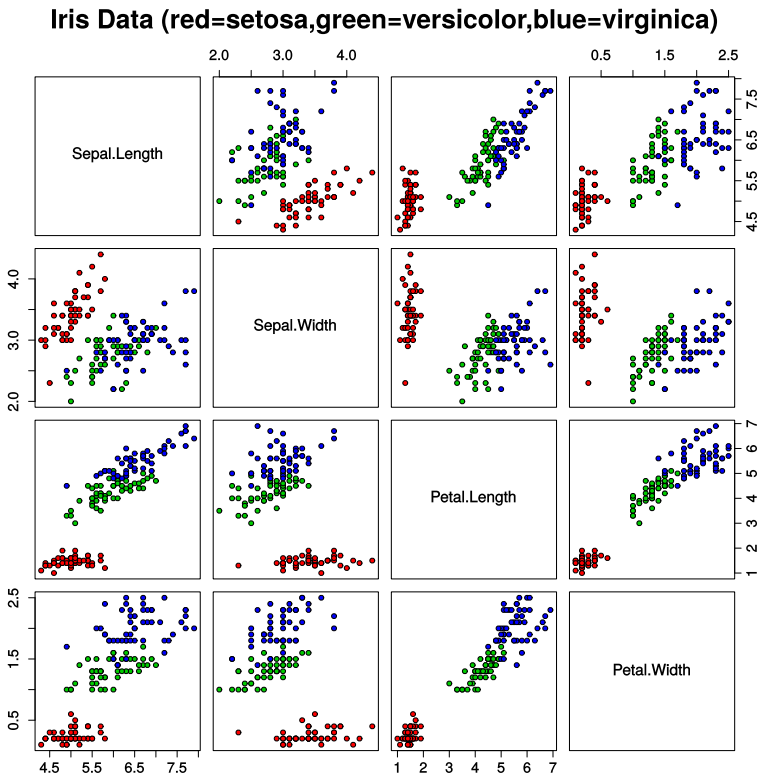

Figure 6: `Iris` dataset[4]

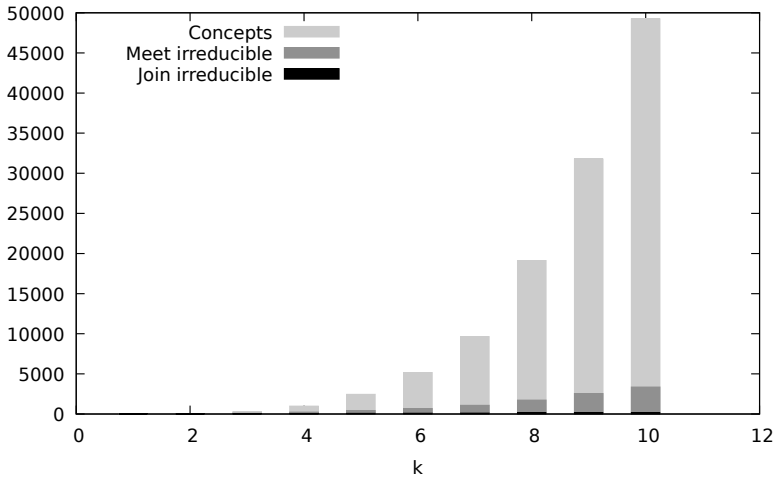

Figure 7: Analysis of the influence of the number of best entropy measures kept for the `Iris` dataset

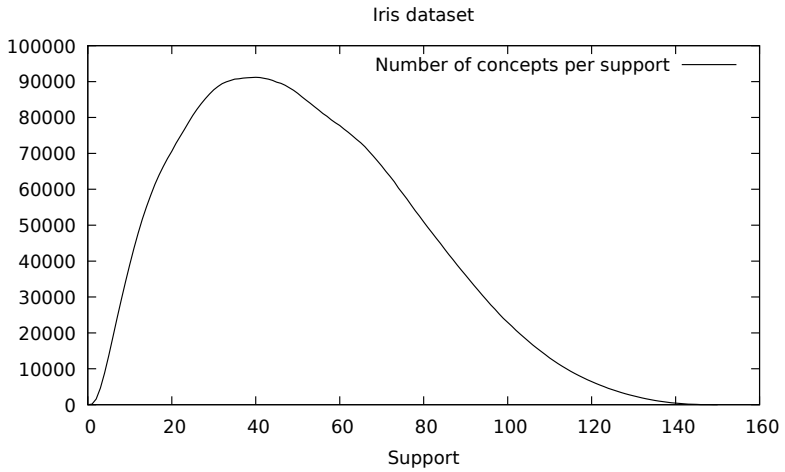

Figure 8: Number of concept per support for the `Iris` dataset

The concept lattice, displayed in Figure 9, is composed of 21 concepts. The last concept which is the absurd one (the extent is an empty set) is described by 2 predicates whose conjunction is always false.

4.2.4.4. ***`Digit`*** *dataset with minimal logical formula as descriptions.* As last example, we consider the `digit` described by their properties **c**omposed, **e**ven, **o**dd, **p**rime and **s**quare given in Table 1.

These 5 characteristics are considered together, and we compute their minimal boolean formulae as description using the Quine-McCluskey algorithm [14]. The strategy consists in trying to add an attribute or its negation at each step.

The concept lattice is displayed in Figure 10. We can observe that the maximum number of predecessors cannot exceed 5 since the predecessors are maximum per inclusion.

5. Conclusion

We have described our NextPriorityConcept algorithm for complex and heterogenous mining using a pattern discovery approach.

More precisely, our algorithm generates formal concepts using the dual version of Bordat's theorem for the generation of immediate predecessors (instead of immediate successors), and where recursion is replaced by a priority queue using the support of concepts to make sure that concepts are generated level by level, each concept being generated before its predecessors. Moreover, a constraint propagation mechanism ensures that meets are correctly generated.

Heterogeneous data are provided at input with a description mechanism and a predecessor generation strategy adapted to each kind of data, and generically described by predicates.

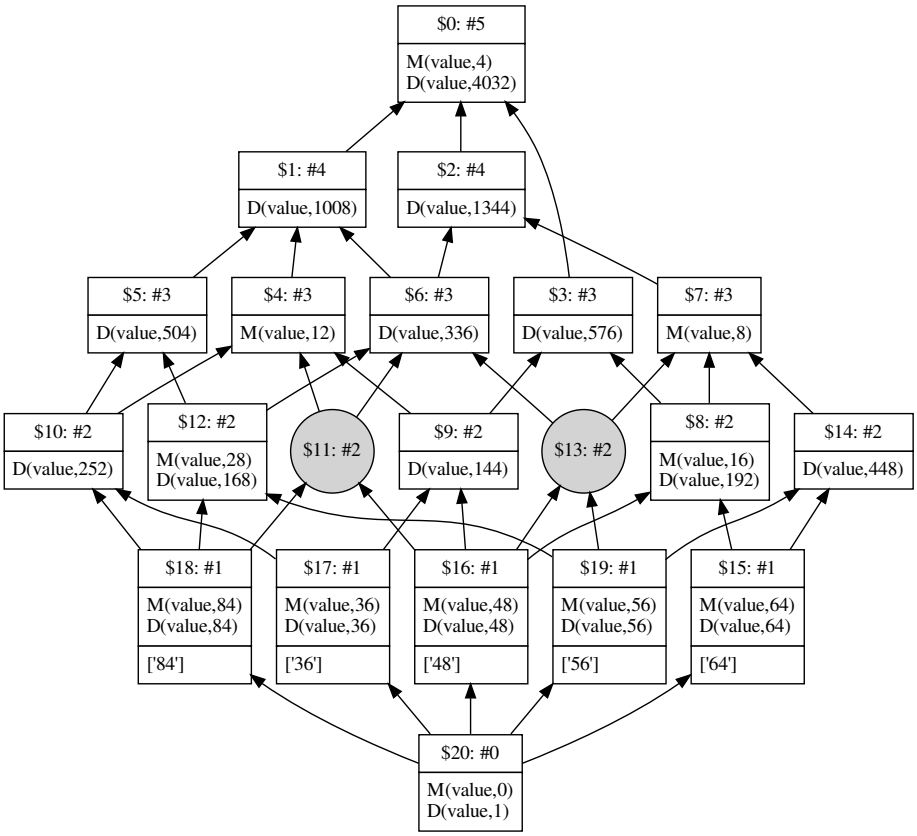

Figure 9: `Numbers` dataset with GCD and LCM as descriptions (M(value, x) represents the fact that the value is a multiple of x and D(value, x) represents the fact that the value is a divisor of x

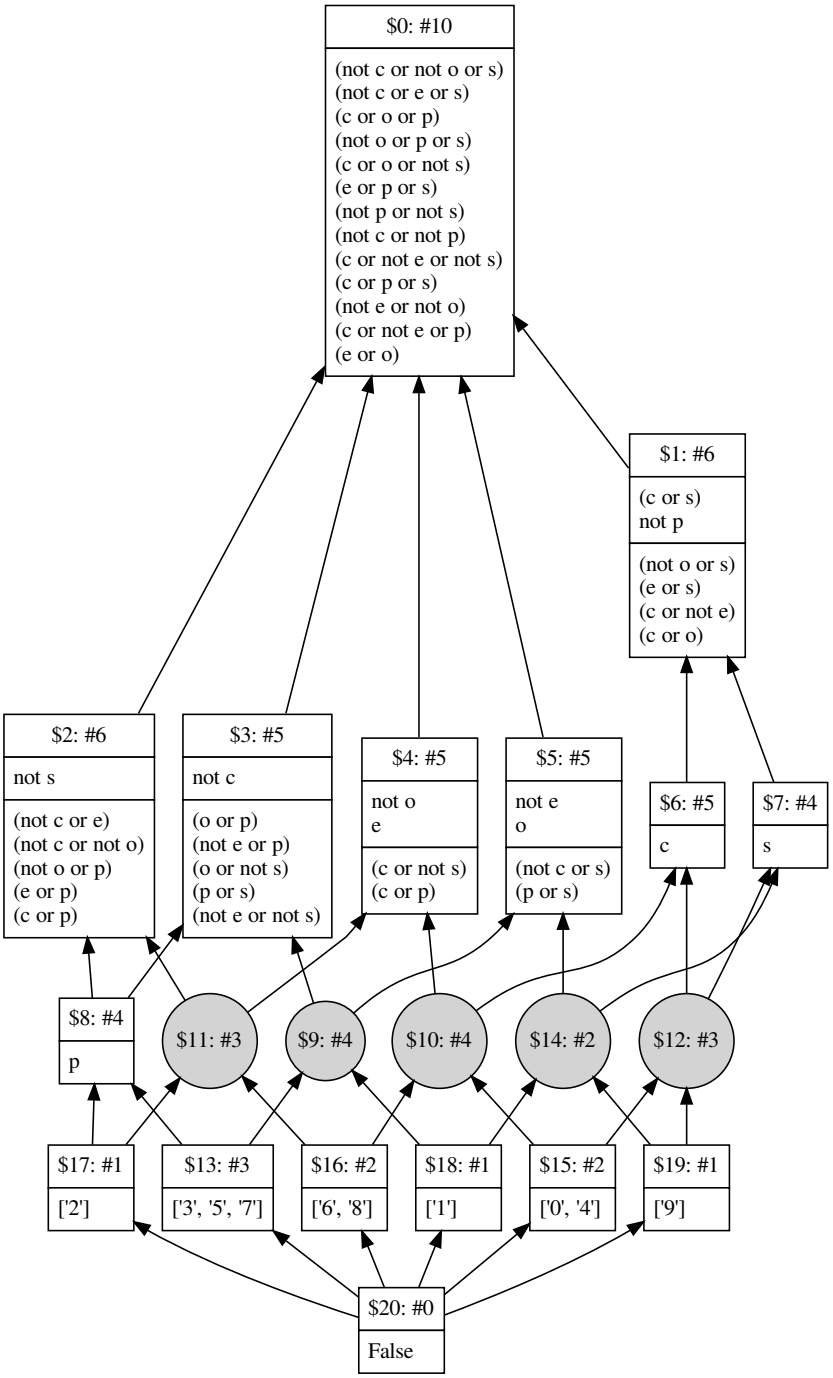

Figure 10: `Digit` dataset with minimal logical formula as description

Our algorithm is generic and agnostic since we use predicates whatever the characteristics. It is implemented with a system of plugins for an easy integration of new characteristics, new description, new strategies and new meta-strategies. The python3 implementation of `GALACTIC` (**GA**lois **LA**ttices, **C**oncept **T**heory, **I**mplicational systems and **C**losures) is available since January 2020.[5]

We have already implemented some descriptions and strategies plugins for boolean, numeric, categorical attributes, strings and sequences. We are currently working on descriptions and strategies for graphs and triadic data.

We plan to study more precisely the theoretical properties of the concept lattice computed. Indeed, this lattice seems to be a suborder of the lattice of the initial context, depending on the strategy. Therefore, it corresponds to a reduction of the concept space, and a better understanding of its properties would make it possible to envisage relevant reduction mechanisms of the search space.

Acknowledgements

Thanks are owed to Rokia Missaoui and Gaël Lejeune for their constructive and helpfull comments. The work of Sergei O. Kuznetsov presented in Sections 1-3 was carried out at St. Petersburg Department of Steklov Mathematical Institute of Russian Academy of Science and supported by the Russian Science Foundation grant no. 17-11-01276.

[1] M. Barbut, B. Monjardet, Ordres et classifications : Algèbre et combinatoire, Hachette, Paris, 1970, 2 tomes.

[2] B. Ganter, R. Wille, Formal Concept Analysis, Mathematical foundations, Springer Verlag, Berlin, 1999.

[3] K. Bertet, Ch. Demko, J.-F. Viaud, C. Guérin, Lattices, closure systems and implication bases: A survey of structural aspects and algorithms, Theoretical Computer Science 743 (2018) 93–109.

[4] B. Ganter, S. O. Kuznetsov, Pattern structures and their projections, in: LNCS of International Conference on Conceptual Structures (ICCS'01), 2001, pp. 129–142.

[5] M. Kaytoue, V. Codocedo, A. Buzmakov, J. Baixeries, S. O. Kuznetsov, A. Napoli, Pattern structures and concept lattices for data mining and knowledge processing, in: In Proceedings of ECML-PKDDl, 2015, pp. 227–231.

[6] M. Kaytoue, Contributions to pattern discovery, Habilitation, University of Lyon, France (february 2020).

[7] S. Ferré, O. Ridoux, A logical generalization of formal concept analysis, Vol. 1867, 2000, pp. 371–384.

[8] S. Ferré, Reconciling expressivity and usability in information access - from filesystems to the semantic web, Habilitation, University of Rennes 1, France (november 2014).

[9] J.-P. Bordat, Calcul pratique du treillis de Galois d'une correspondance, Mathématiques et Sciences humaines 96 (1986) 31–47.

[10] C. Linding, Fast concept analysis, in: Working with Conceptual Structures-Contributions to ICC, 2002, pp. 235–248.

[11] S. Eddine Boukhetta, Ch. Demko, J. Richard, K. Bertet, Sequence mining using FCA and the NextPriorityConcept algorithm, in: Concept Lattice and Applications (CLA'20), 2020.

[12] A. Belfodil, S. O. Kuznetsov, C. Robardet, M. Kaytoue, Mining Convex Polygon Patterns with Formal Concept Analysis, in: C. Sierra (Ed.), The Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI), Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence Main track, Melbourne, Australia, 2017, pp. 1425–1432. doi:10.24963/ijcai.2017/197. URL https://hal.archives-ouvertes.fr/hal-01573841

[5] https://galactic.univ-lr.fr

[13] B. Chazelle, An optimal convex hull algorithm in any fixed dimension, Discrete and Computational Geometry 10 (1) (1993) 377–409.
[14] W. V. O. Quine, The problem of simplifying truth functions, The American Mathematical 59 (8) (1952) 521–531.

Data:
- $\langle G, M, (\alpha, \beta)\rangle$ a formal context
- σ a strategy

Output:
- the formal context $\langle G, P, I_P\rangle$
- its concepts (A, D)

begin

```
/* Priority queue for the concepts */
Q ← [] ;      /* Q is a priority queue using the support of concepts */
Q.push((|G|, (G, α(G)))) ;                  /* Add the top concept into Q */

/* Data structure for constraints */
C ← [] ;   /* C is the descendant constraints map being ∅ by default */

/* Data structures for new attributes */
P ← ∅ ;                          /* P is the set of selected attributes */
I_P ← ∅ ;             /* I_P is the new binary relation between G and P */

/* Immediate predecessors generation */
while Q not empty do
    /* Compute concept */
    (A, D) ← Q.pop() ;           /* Get the concept with highest support */
    produce (A, D) ;

    LP ← Predecessors-Strategy((A, D), P, I_P, C, σ) ;

    /* Update queue */
    forall the (A', D') ∈ LP do
        if (A', D') ∉ Q then
            Q.push((|A'|, (A', D'))) ;              /* Add new concept into Q */
        end
    end

    delete C[A] ;                                    /* Remove useless data */
end
return ⟨G, P, I_P⟩
```

end

Algorithm 4: NextPriorityConcept-Strategy

Data:
- (A, D) a concept
- P the set of selected attributes
- I_P the new relation
- $\mathcal{C}$ the constraints
- σ a strategy

Result:
- LP a set of predecessors

begin
  $L \leftarrow \emptyset$;
  forall the $b \in (\sigma(A) \cup \mathcal{C}[A]) \setminus D$ **do**
    /* b is a new "potential" attribute for a predecessor */
    $A' \leftarrow \beta(b) \cap A$; /* Compute the objects of $D \cup \{b\}$ */
    /* Add (A', b) if A' maximum in L and included in A */
    if $A' \subset A$ **then** $L \leftarrow$ Inclusion-Max(L,(A', b));
  end
  $N \leftarrow \{b : (A', b) \in L\}$; /* N is the set of new constraints */
  $LP \leftarrow \emptyset$;
  forall the $(A', b') \in L$ **do**
    /* Update the selected attributes P and the relation I_P*/
    if $b' \in \sigma(A)$ **then**
      $P \leftarrow P \cup \{b'\}$; /* Update the set of selected attributes */
    end
    $D' \leftarrow \alpha(A') \cap P$; /* Compute the extent of A' */
    LP.add((A', D')) ; /* (A', D') is a new concept */
    $I_P \leftarrow I_P \cup (A' \times D')$; /* Update the new relation */
    /* Compute residual and cross constraints */
    $\mathcal{C}[A'] \leftarrow \mathcal{C}[A'] \cup \mathcal{C}[A] \cup N \setminus D'$
  end
  return LP
end

Algorithm 5: Predecessors-Strategy

Data:

- $\langle G, S\rangle$ a dataset
- $(S^i)_{i\leq d}$ a family of S
- δ a description
- σ a strategy

Output:

- the formal context $\langle G, P, I_P\rangle$
- its concepts (A, D)

begin

```
/* Priority queue for the concepts */
Q ← [] ;      /* Q is a priority queue using the support of concepts */
Q.push((|G|, (G, δ(G))) ;                    /* Add the top concept into Q */

/* Data structure for constraints */
C ← [] ;   /* C is the descendant constraints map being ∅ by default */
/* Data structures for predicates */
P ← ∅ ;                                  /* P is the set of all predicates */
I_P ← ∅ ;                 /* I_P is the binary relation between G and P */
/* Immediate predecessors generation */
while Q not empty do
    (A, D) ← Q.pop() ;            /* Get the concept with highest support */
    produce (A, D) ;
    LP ← Predecessors-Desc((A, D), P, I_P, C, σ, δ) ;
    /* Update queue */
    forall the (A', D') ∈ LP do
        if (A', D') ∉ Q then
            Q.push((|A'|, (A', D'))) ;                  /* Add concept into Q */
        end
    end
    delete C[A] ;                                       /* Remove useless data */
end
return ⟨G, P, I_P⟩
```

end

Algorithm 6: NextPriorityConcept

Data:

- (A, D) a concept
- P the set of predicates
- I_P the binary relation between G and P
- $\mathcal{C}$ the constraints
- σ a strategy (issued from the σ^i)
- δ a description (issued from the δ^i)

Result:

- LP a set of predecessors

begin

$L \leftarrow \emptyset$;
forall the $p \in (\sigma(A) \cup \mathcal{C}[A]) \setminus D$ **do**
/* p is a new "potential" selector to generate a predecessor */
$A' \leftarrow \{a \in A \ : \ p(a)\}$; /* A' are the objects verifying $D \cup \{p\}$ */
/* Add (A', p) if A' maximum in L and included in A */
if $A' \subset A$ **then** $L \leftarrow$ Inclusion-Max(L,(A', p));
end
$N \leftarrow \{p \ : \ (A', p) \in L\}$; /* N is the set of new constraints */
$LP \leftarrow \emptyset$;
forall the $(A', p') \in L$ **do**
/* Update the selected attributes P and the relation I_P*/
if $p' \in \sigma(A)$ **then**
$P \leftarrow P \cup \{p'\}$; /* Update the set of selected predicates */
end
$D' \leftarrow \delta(A')$; /* D' are the new predicates describing A' */
LP.add((A', D')) ; /* (A', D') is a new concept */
$I_P \leftarrow I_P \cup (A' \times D')$; /* Update the new relation */
/* Compute cross constraints (X) and propagate constraints */
$X \leftarrow \{p'' \in N \ : \ p''(a) \ \forall a \in A'\}$
$\mathcal{C}[A'] \leftarrow \mathcal{C}[A'] \cup \mathcal{C}[A] \cup N \setminus X$
end
return LP

end

Algorithm 7: Predecessors-Desc