Enumerating Pseudo-Intents in a Partial Order

Alexandre Bazin and Jean-Gabriel Ganascia

Université Pierre et Marie Curie, Laboratoire d'Informatique de Paris 6
Paris, France
Alexandre.Bazin@lip6.fr
Jean-Gabriel@Ganascia.name

Abstract. The enumeration of all the pseudo-intents of a formal context is usually based on a linear order on attribute sets, the lectic order. We propose an algorithm that uses the lattice structure of the set of intents and pseudo-intents to compute the Duquenne-Guigues basis. We argue that this method allows for efficient optimizations that reduce the required number of logical closures. We then show how it can be easily modified to also compute the Luxenburger basis.

1 Introduction

Various fields, such as artificial intelligence, data mining and databases, are concerned with the problem of finding implications between sets of attributes describing objects. Formal concept analysis offers a sound mathematical framework to help develop algorithms that compute implications. As it has been shown in [3], the set of implications of minimum cardinality needed to obtain all the interesting information can be found by enumerating attribute sets known as pseudo-intents. The best-known algorithm for the computation of pseudo-intents is Next Closure [4], in which pseudo-intents are enumerated following a total order on attribute sets called lectic order. The problem of enumerating pseudo-intents in lectic order has been studied in [2] and found to be CONP-complete. The same work has been unable to find a lower bound to the complexity of this problem once the restriction on the order is removed. This indicates that the order plays an important role and that using a weaker order could potentially yield interesting results. This prompted us to study an algorithm based on the same principles as Next Closure but with a partial order on the attribute sets instead of a linear order. We are inspired by the multiple algorithms for the related problem of enumerating formal concepts that make use of the lattice structure of the problem. There already are approaches that do not make use of the lectic order for pseudo-intents, such as the attribute incremental algorithm that has been introduced in [8], or the divide-and-conquer approach in [9] but we will not compare it to our own in this work as their nature differs.

After an overview of the notations in formal concept analysis and a brief recall of Next Closure, we will present our algorithm. In a second part, we will show that it can be modified to compute not only the Duquenne-Guigues basis but also the Luxenburger basis. The functioning of the algorithm will then be illustrated by means of a small example.

ISBN 978–2–7466–6566–8, Laboratory L3i, University of La Rochelle, 2013.

2 Notations

In formal concept analysis, data is represented by a triple $\mathcal{C} = (\mathcal{O}, \mathcal{A}, \mathcal{R})$, called *formal context*, where $\mathcal{O}$ is a set of objects, $\mathcal{A}$ a set of binary attributes and $\mathcal{R} \subseteq \mathcal{O} \times \mathcal{A}$ a relation assigning a set of attributes to each object. An object o is said to be *described* by an attribute set A iff $(o, a) \in \mathcal{R}$ for every $a \in A$.

Two derivation operators are commonly employed and defined as follows :

$$.' : 2^{\mathcal{A}} \to 2^{\mathcal{O}}$$

$$A' = \{o \in \mathcal{O} \mid \forall a \in A, (o, a) \in \mathcal{R}\} \tag{1}$$

$$.' : 2^{\mathcal{O}} \to 2^{\mathcal{A}}$$

$$O' = \{a \in \mathcal{A} \mid \forall o \in O, (o, a) \in \mathcal{R}\} \tag{2}$$

The compositions of these two operators are closure operators and we will use the notation $.''$ for both. A set A is said to be *closed* if $A = A''$. A closed attribute set is called an *intent* and a closed object set an *extent*. A pair (E, I) where $E \subseteq \mathcal{O}$ and $I \subseteq \mathcal{A}$ is called a *formal concept* of the context if $I = E'$ and $E = I'$. A formal concept (A, B) is said *more general* than a concept (C, D) if $C \subset A$ or $B \subset D$. The set of all concepts of a given context ordered in this way is a complete lattice called *concept lattice* of the context.

An *implication* is a relation between two attribute sets A and B, denoted by $A \to B$. An implication $A \to B$ is said to *hold* in a context if every object described by A is also described by B. We write $A \equiv B$ if $A \to B$ and $B \to A$. The implication $A \to A''$ always holds in the context.

An attribute set A is a *pseudo-intent* if it is not an intent and $P'' \subset A$ for all pseudo-intents $P \subset A$. The set $\mathcal{B} = \{A \to A'' \mid A \text{ is a pseudo-intent}\}$, called the *canonical basis*, is a set of implications of minimum cardinality such that every implication that holds in the context can be inferred from it through the application of the Armstrong rules. That is, it is the minimum number of implications containing all information on the relations between attribute sets in the context.

Computing the canonical basis of a context thus requires the enumeration of its pseudo-intents. This presents some challenges as the number of pseudo-intents can be exponential in the size of the context (the number of attributes), as shown in [5]. Moreover, whether pseudo-intents can be enumerated without intents is still an open problem and the number of intents can itself be exponential in the number of pseudo-intents.

2.1 Next Closure

The most known algorithm for computing the canonical basis is Ganter's Next Closure [4]. For an arbitrary linear order $<$ on $\mathcal{A}$, a linear order, called *lectic order*, $\leq_{lec}$ on attribute sets is defined as follows:

$$A \leq_{lec} B \Leftrightarrow \min(A \Delta B) \in B$$

Δ is the symmetric difference. The implicational closure of an attribute set A with respect to a set of implications $\mathcal{L}$, usually noted $\mathcal{L}(A)$, is the smallest superset of A such that, for every $X \subseteq \mathcal{L}(A)$, we have $X \to Y \in \mathcal{L} \Rightarrow Y \subseteq \mathcal{L}(A)$. The implicational closure is a closure operator. The implicational pseudo-closure of an attribute set A, noted $\mathcal{L}^-(A)$, is the smallest superset of A such that, for every $X \subset \mathcal{L}(A)$, we have $X \to Y \in \mathcal{L} \Rightarrow Y \subseteq \mathcal{L}(A)$.

It is shown that both intents and pseudo-intents are closed under $\mathcal{B}^-$. Next Closure, in its original form, computes closed sets for a given closure operator. Applied to the computation of pseudo-intents, it goes through implicationally pseudo-closed subsets of $\mathcal{A}$ in lectic order and checks whether the result is an intent or a pseudo-intent. The lectic order being linear, every set is computed once and only once. For an attribute set A and an attribute i, the operator $\oplus$ is defined as follows :

$$A \oplus i = \mathcal{B}^-(\{a \in A \mid a \leq i\} \cup \{i\})$$

The closed set that immediately follows A in the lectic order is $A \oplus i$ with i maximal such that $\min((A \oplus i) \setminus A)$. For each set closed under $\mathcal{B}^-$, Next Closure tests whether it is an intent or a pseudo-intent and then constructs the next set with the operator $\oplus$ until it reaches $\mathcal{A}$. Numerous optimizations can be added to reduce the number of implicational closures. For example, as proposed in [8], if $Next(A) = A \oplus i = B$ and $\min(B'' \setminus A) < \max(A)$ then we can continue as if $A \oplus i$ had been rejected by Next Closure. It is the only one we will consider here as it is specific to Next Closure and has no incidence on our algorithm.

3 Algorithm

3.1 Principles

We consider the lattice $\Phi = (\{A = \mathcal{B}^-(A) \mid A \subseteq \mathcal{A}\}, \subseteq)$ of attribute sets closed under $\mathcal{B}^-$ ordered by the inclusion relation. Enumerating pseudo-intents is enumerating all the elements of Φ and testing whether they are intents or pseudo-intents. As with other enumeration problems, we want to make sure every element is found once and only once. The other great FCA problem, computing the formal concepts of a context, can be viewed as the enumeration of the elements of $(\{\mathcal{B}(A) \mid A \subseteq \mathcal{A}\}, \subseteq)$. It has been extensively studied (see [6] for a comparison of different algorithms) and various approaches have been proposed for both generating elements from previous ones and checking whether an attribute set has already been found (or even preventing the generation of redundant sets).

Surprisingly, it seems only Next Closure is commonly used for the problem of pseudo-intents, non-batch algorithms excluded. Indeed, it is very efficient in that the lectic order on attribute sets allows for the generation of the next set without stocking more than the current set and the total order ensures we do not generate them more than once. However, we feel that the fact that sets are not always generated by one of their subsets is a weakness for some applications, such as data mining. We propose to modify Next Closure with techniques from some algorithms for formal concepts that make use of the lattice structure.

The lectic order has the interesting property that a set B is greater than all its subsets. If $B \in \Phi$ is supposed to be generated by a single set, it should be one that is easily identifiable such as its lectically greatest subset $A \in \Phi$. For any two A and B in Φ, we shall use $A \rightsquigarrow B$ to denote the fact that A is the lectically greatest subset of B closed under $\mathcal{B}^-$ or, equivalently, that B is generated from A. We say that A is the predecessor of B and B is a successor of A.

Proposition 1. *The relation $\rightsquigarrow$ defines a spanning tree of the neighbouring graph of Φ.*

Proof. Any non-empty set B has at least one strict subset in Φ. We call A its lectically greatest subset in Φ. The lectic order respects the inclusion relation so there is no C such that $A \subset C \subset B$ and A is a lower cover of B. Since every non-empty $B \in \Phi$ has a single lower cover A such that $A \rightsquigarrow B$, the successor relation defines a spanning tree of the neighbouring graph of Φ. □

Proposition 2. *For any two A, $B \in \Phi$, the attribute set B is an upper cover of A if and only if $B = \mathcal{B}^-(A \cup \{i\})$ for every $i \in B \setminus A$.*

Proof. If B is an upper cover of A, then there is no C such that $A \subset \mathcal{B}^-(C) \subset B$. We have $\mathcal{B}^-(A \cup \{i\}) \subseteq B$ when $i \in B \setminus A$, so $B = \mathcal{B}^-(A \cup \{i\})$ for all $i \in B \setminus A$.

If $B = \mathcal{B}^-(A \cup \{i\})$ for every $i \in B \setminus A$, given that $A \cup \{i\}$ is the smallest subset of B that contains both A and i, then there is no superset of A that is a strict subset of B. □

Attribute sets are generated according to the tree, starting from $\emptyset$. The recursive nature of the definition of a pseudo-intent prevents us from computing a set before the computation of all its subsets. The lectic order solves this problem efficiently but our tree contains only the knowledge of the lectically greatest subset. In order to make sure that every subset has been found first, we propose to consider attribute sets in order of increasing cardinality. This way, when an intent or pseudo-intent of cardinality n is considered, we are sure that all pseudo-intents of cardinality $n - 1$ have been found.

Proposition 3. *If $i < \min(A)$ and $A \subset A \oplus i$, then $A \oplus i$ has a subset in Φ that is lectically greater than A.*

Proof. If $i < \min(A)$, then $\mathcal{B}^-(\{a \in A \mid a \leq i\} \cup \{i\}) = \mathcal{B}^-(\{i\})$. If $A \subset A \oplus i$, then $\{i\}$ is a pseudo-intent that is a subset of $A \oplus i$ and is lectically greater than A. □

Algorithm 1 Successors(A)

```
1: Successors = ∅
2: for every attribute i greater than min(A) do
3:    B = A ⊕ i
4:    if A ⊂ B then
5:       if i = min(B \ A) and ∀j ∈ B \ A, A ⊕ j = B then
6:          Successors = Successors ∪ {B}
7:       end if
8:    end if
9: end for
10: return Successors
```

Proposition 4. *For any two attribute sets A and B such that $A \subset B$, $A \rightsquigarrow B \Leftrightarrow \forall i \in B \setminus A, A \oplus i = B$.*

Proof. If $\forall i \in B \setminus A, A \oplus i = B$, then B has no subset in Φ lectically greater than A. As such, $\forall i \in B \setminus A, A \oplus i = B \Rightarrow A \rightsquigarrow B$.

If $A \rightsquigarrow B$ and $\exists i \in B \setminus A$ such that $A \oplus i \subset B$, then B has a subset in Φ lectically greater than A, which contradicts the hypothesis. As such, $A \rightsquigarrow B \Rightarrow \forall i \in B \setminus A, A \oplus i = B$. □

So, in order to generate all the successors of A, it suffices to compute $A \oplus i$ for every i greater than $\min(A)$. If a resulting attribute set is an upper cover of A it is a successor.

Testing whether a new attribute set $B = A \oplus i$ is a successor of A is easy as we know $A \oplus j$ for every $j \in B \setminus A$. Algorithm 1 uses this to compute the successors of an attribute set.

If an attribute set B is a pseudo-intent, it is a $\wedge$-irreducible element of Φ. That is, it has a single upper cover. If $B = \mathcal{B}^-(A)$, the set B'' is the lectically next set after B if $\max(A) \leq \min(A'' \setminus A)$. That way, if an attribute set is a pseudo-intent, we do not have to explicitly compute its successors.

Algorithm 2 uses Algorithm 1 to go through the intents and pseudo-intents of the context and compute its canonical basis. It starts with $\emptyset$ and computes the closure of all the attribute sets of the same cardinality simultaneously before generating their successors. However, when an intent A is considered and its successors generated, only the pseudo-intents of cardinality lesser than $|A| + 1$ have been found so the $A \oplus i$ computed at this point might be different from the final $A \oplus i$. For example, if $|A \oplus i| - |a| = 2$, an implication $B \to C$ with $|B| = |A| + 1$ and $B \subset A \oplus i$ will change the result. For this reason, we must split Algorithm 1 into two parts. First, the sets $A \oplus i$ are computed for all i once every intent of cardinality $|A|$ have been found. Then, when $A \oplus i$ is considered as a Candidate, we must check whether it is still logically closed and, if it is, test whether it is a successor with Proposition 4.

Algorithm 2 Computing the Duquenne-Guigues Basis

$Card = 0$
$Candidates = \{\emptyset\}$
$\mathcal{B} = \emptyset$
while $Candidates \neq \emptyset$ **do**
  $Intents = \emptyset$
  for every attribute set A of cardinality $Card$ in $Candidates$ **do**
    if $A = \mathcal{B}^-(A)$ and it is a successor of its generator **then**
      $B = A''$
      if $A \neq B$ **then**
        $\mathcal{B} = \mathcal{B} \cup \{A \rightarrow B\}$
        if B lectically follows A **then**
          $Candidates = Candidates \cup \{B\}$
        end if
      else
        $Intents = Intents \cup \{A\}$
      end if
    else
      $A = \mathcal{B}^-(A)$
    end if
  end for
  for every set C in $Intents$ **do**
    $Candidates = Candidates \cup \{C \oplus i \mid i \in \mathcal{A} \text{ and } i = \min((C \oplus i) \setminus C)\}$
  end for
  $Card = Card + 1$
end while
return $\mathcal{B}$

Proposition 5. *Algorithm 2 terminates and produces the canonical basis of the context.*

Proof. An attribute set can only be used to generate attribute sets of greater cardinality. The set of attributes is finite, so the algorithm terminates when it reaches $\mathcal{A}$.

Every element of Φ, except $\emptyset$, has a lower neighbour that is a predecessor in the spanning tree. If we generate the attribute sets along the tree, the lattice Φ being complete, every one of its elements is considered at least once. If an attribute set A is not equal to $\mathcal{B}^-(A)$ it is not a pseudo-intent so all pseudo-intents are found. □

During Algorithm 2, Algorithm 1 is applied to every intent to generate attribute sets. Their closure is then computed. As such, Algorithm 2 is in $\mathcal{O}(|\Phi| \times (X + Y))$ where X is the complexity of computing the closure of a set and $Y = |A| \times L$ is the complexity of generating successors that depends on the complexity L of the saturation algorithm used. A study of algorithms for computing the implicational closure and their use in our problem can be found in [1]. Note that, since every potential pseudo-intent P is constructed from an

intent I with $I \subset P$, computing P'' can be done on the subcontext containing the objects of I'.

3.2 Improvements

Additional reduction of the number of implicational closures required to compute the base can be achieved by making use of the property that every attribute set is constructed by adding an attribute to one of its subsets.

Proposition 6. *If $B = A \oplus i$ is a successor of A with $i = \min(B \setminus A)$, then $B \oplus j = A \oplus j$ for every attribute $j < i$.*

Proof. If $j < i$, then $\mathcal{B}^-(\{b \in B \mid b < j\} \cup \{j\}) = \mathcal{B}^-(\{a \in A \mid a < j\} \cup \{j\})$ because $i = \min(B \setminus A)$. Thus, we have $B \oplus j = A \oplus j$. □

This lets us use the knowledge acquired on a set A to reduce the number of necessary logical closures on its supersets. Indeed, for any $B = A \oplus i$ with $i = \min(B \setminus A)$, the set $B \oplus j$ is already known for every attribute lesser than i. This means that, in order to compute the successors of B, we need only to use the $\oplus$ operator with attributes greater than i.

We can also use the fact that there are pseudo-intents P such that $P'' = \mathcal{A}$, which means that no object of the context is described by P. Indeed, for an intent A such that $A \rightsquigarrow A \oplus i$, there are two possibilities for i. If $i \in \bigcup_{o \in A'} \{o\}'$, meaning there is at least an object described by $A \cup \{i\}$, the set $A \oplus i$ is either an intent or a pseudo-intent. If $i \in \mathcal{A} \setminus \bigcup_{o \in A'} \{o\}'$, meaning no object is described by $A \cup \{i\}$, the closure of $A \oplus i$ is $\mathcal{A}$ and, for any superset B of A, the set $B \oplus i$ is either equal to $A \oplus i$ or $\mathcal{A}$. This means that computing $B \oplus i$ is unnecessary for every successors B of A in the spanning tree.

4 Computing a Base for Partial Implications

4.1 Luxenburger Basis

A partial implication between two attribute sets A and B, otherwise called association rule, is a relation of the form $A \to_{s,c} B$ where s is called the *support* and c the *confidence*. It means that s% of the objects of the context are described by A and that c% of them are also described by B. Implications, as defined in Section 2, are partial implications with a confidence of 100%. An attribute set A having the same support as A'', the confidence and support of a partial implication $A \to_{s,c} B$ can be derived from $A'' \to_{s,c} B''$. As such, to obtain a basis for partial implications, one needs only to know the intents of the formal context.

It has been shown in [7] that a minimal basis for partial implications, called Luxenburger basis, is obtained by considering a set $\{A \to_{s,c} B \mid A = A'' \text{ and } B = B''\}$ that forms a spanning tree of the neighbouring graph of the intent lattice such that $\mathcal{A}$ is the conclusion of a single partial implication.

4.2 Modification of the Algorithm

In Algorithm 2, every attribute set is generated from a single intent with the exception of some essential intents that lectically follow a pseudo-intent. We would like those intents to be constructed from their lectically greatest subset that is an intent instead of just their lectically greatest subset. Let us suppose that we have an intent A and a pseudo-intent B such that $A \rightsquigarrow B$ and $B \rightsquigarrow B''$. The lectically greatest intent that is a subset of B'' is either A or a superset of A so it can be constructed by adding attributes of $B'' \setminus A$ to A.

Algorithm 3 computes C for a given A, B and B''.

Algorithm 3 Lectically Greatest Sub-Intent

```
1: C = A
2: for every attribute i in B'' \ A in increasing order do
3:    if B(C ∪ {i}) ≠ B'' then
4:       C = C ∪ {i}
5:    end if
6: end for
7: return C
```

Proposition 7. *Algorithm 3 terminates and computes the lectically greatest intent that is a subset of B''*

Proof. There is a finite number of attributes and the loop goes through them in a total order so the algorithm terminates. The attribute set it returns, C, is either A or closed under $\mathcal{B}(.)$, so it is an intent. It is the implicational closure of a subset of B'' and it is not equal to B'' so we have $C \subset B''$. Let us suppose that there is an intent $D \subset B''$ lectically greater than C with $i = \min(C \Delta D)$. The attribute i is such that $\mathcal{B}(X \cup \{i\}) \subset B''$ for any $A \subseteq X \subseteq D$ so the algorithm must have added it to C and we have $C = D$.

Thus, the algorithm returns C, the greatest intent that is a subset of B''. □

If, in Algorithm 2, we maintain the spanning tree we used to generate attribute sets, it is easy to apply Algorithm 3 to every intent that lectically follows a pseudo-intent. If we change the predecessor of those intents to the attribute set computed by Algorithm 3 we obtain a spanning tree of Φ in which the predecessor of every intent is an intent. As $\mathcal{A}$ also has a unique predecessor, it gives us the Luxenburger basis of the context along with the Duquenne-Guigues basis.

5 Example

We apply Algorithm 2 on the following context with $a < b < c < d < e$:

	a	b	c	d	e
o1	×	×			
o2		×		×	×
o3		×	×	×	
o4			×		×
o5				×	×

Fig. 1. Context 1

The algorithm starts with $\emptyset$. It is closed so we generate its successors.

$$\left|\begin{array}{l}\emptyset \oplus e = \mathbf{e}\\ \emptyset \oplus d = \mathbf{d}\\ \emptyset \oplus c = \mathbf{c}\\ \emptyset \oplus b = \mathbf{b}\\ \emptyset \oplus a = \mathbf{a}\end{array}\right|$$

The set of $Candidates$ is then $\{a, b, c, d, e\}$. Among them, only a is a pseudo-intent with $a'' = ab$ so we add $a \to ab$ to the basis. Moreover, ab lectically follows a so we add ab to $Candidates$. Its lectically greatest subset that is an intent is b. We then generate the successors of b, c, d and e. There are no attributes greater than $\min(e) = e$ so e has no successors.

$$\left|\begin{array}{l}b \oplus e = \mathbf{be}\\ b \oplus d = \mathbf{bd}\\ b \oplus c = \mathbf{bc}\end{array}\right| \left|\begin{array}{l}c \oplus e = \mathbf{ce}\\ c \oplus d = \mathbf{cd}\end{array}\right| \left|\begin{array}{l}d \oplus e = \mathbf{de}\end{array}\right|$$

The set of $Candidates$ is then $\{ab, bc, bd, be, cd, ce, de\}$. Among them, bc, be and cd are pseudo-intents so we add $bc \to bcd$, $be \to bde$ and $cd \to bcd$ to $\mathcal{B}$. The set bcd lectically follows bc so we add bcd to $Candidates$. Its lectically greatest subset that is an intent is bd. We then generate the successors of ab, bd, ce and de. The set de has no successors for the same reasons as the set e in the previous step and we already know that $c \oplus d = cd$ so $ce \oplus d$ is known and ce has no successors.

$$\left|\begin{array}{l}ab \oplus e = abde\\ ab \oplus d = \mathbf{abd}\\ ab \oplus c = abcd\end{array}\right| \left|\begin{array}{l}bd \oplus e = \mathbf{bde}\end{array}\right|$$

The set of $Candidates$ is then $\{abd, bcd, bde\}$. Among them, abd is a pseudo intent so we add $abd \to abcde$ to $\mathcal{B}$. The set $abcde$ lectically follows abd so we add $abcde$ to $Candidates$. Its lectically greatest subset that is an intent is ab. We then generate the successors of bcd and bde. We already know $bd \oplus c$ so we also know $bde \oplus c$. As such, computing $bde \oplus c$ is not needed and no other attribute is available so bde has no successors.

$$\left|bcd \oplus e = \mathbf{bcde}\right|$$

The set of $Candidates$ is then $\{bcde, abcde\}$. Only $bcde$ has a cardinality of 4 and it is a pseudo-intent so we add $bcde \rightarrow abcde$ to $\mathcal{B}$. There are no new intents so we continue with the elements of $Candidates$ of cardinality 5. The set $abcde$ is an intent and has no successors since it is equal to $\mathcal{A}$.

The algorithm stops having produced the following implications :

- $a \rightarrow ab$
- $bc \rightarrow bcd$
- $be \rightarrow bde$
- $cd \rightarrow bcd$
- $abd \rightarrow abcde$
- $bcde \rightarrow abcde$

This is the Duquenne-Guigues basis of the context. In addition, the following spanning tree of the intent lattice has been constructed :

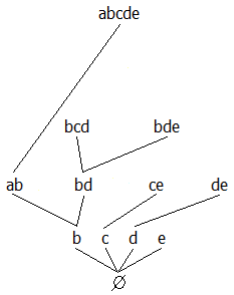

Fig. 2. Spanning Tree of the Lattice of Intents of Context 1

It corresponds to the following set of partial implications :

- $\emptyset \rightarrow_{1,0.6} b$
- $\emptyset \rightarrow_{1,0.4} c$
- $\emptyset \rightarrow_{1,0.6} d$
- $\emptyset \rightarrow_{1,0.6} e$
- $b \rightarrow_{0.6,0.33} ab$
- $b \rightarrow_{0.6,0.66} bd$
- $c \rightarrow_{0.4,0.5} ce$
- $d \rightarrow_{0.6,0.66} de$
- $ab \rightarrow_{0.2,0} abcde$
- $bd \rightarrow_{0.4,0.5} bcd$
- $bd \rightarrow_{0.4,0.6} bde$

To compute it, Algorithm 3 has been used a total of 3 times.

To compute the Duquenne-Guigues basis of this context, our algorithm has performed 16 logical closures. Since the version of Next Closure using the optimizations presented in Section 2.1 would have performed only 15, it is more efficient in this case. However, on the context presented in Figure 3, our algorithm needs only 16 logical closures while Next Closure performs 19 of them. This is due to the fact that attributes are not considered if they have been used to generate an implication $P \to \mathcal{A}$ and this context has a number of pairs of attribute that never appear together. This leads us to believe that our algorithm may be more efficient in those kinds of cases.

	a	b	c	d	e
o1	×				
o2		×			
o3	×	×			
o4					×
o5				×	
o6				×	×
o7			×		
o8			×		×
o9			×	×	
o10			×	×	×

Fig. 3. Context 2

6 Conclusion

We proposed an algorithm that computes the Duquenne-Guigues basis of a formal context. It makes use of the lattice structure of the set of both intents and pseudo-intents in a fashion similar to that of algorithms for computing the concept lattice. The construction of attribute sets is inspired by Next Closure, the most common batch algorithm for computing pseudo-intents, in that it uses the lectic order to ensure the uniqueness of generated sets while avoiding going through what has already been computed. We showed that the algorithm can easily be modified, without much loss complexity-wise, to produce both the Duquenne-Guigues and the Luxenburger bases. Those two bases together are sought after in the domain of association rules mining where it is crucial to obtain a minimal number of rules. They are usually derived from the set of frequent concepts that has to be computed first and, to the best of our knowledge, no algorithm is able to compute them both directly and at the same time without a significant increase in complexity.

Some improvements can be made on the generation of the attribute sets. Most notably, the inclusion test in Algorithm 1 could be done during the computation

of the implicational closure and multiple closures could be realized simultaneously. The fact that attribute sets are effectively generated following a partial order instead of a total order could also permit some degree of parallelization. Such optimizations will be the subject of further research on our part.

References

1. Konstantin Bazhanov and Sergei A. Obiedkov. Comparing performance of algorithms for generating the Duquenne-Guigues basis. In *CLA*, pages 43–57, 2011.
2. Felix Distel and Barış Sertkaya. On the complexity of enumerating pseudo-intents. *Discrete Applied Mathematics*, 159(6):450 – 466, 2011.
3. V. Duquenne and J.-L. Guigues. Famille minimale d'implications informatives résultant d'un tableau de données binaires. *Mathématiques et Sciences Humaines*, 24(95):5–18, 1986.
4. Bernhard Ganter. Two basic algorithms in concept analysis. In *Proceedings of the 8th international conference on Formal Concept Analysis*, ICFCA'10, pages 312–340, Berlin, Heidelberg, 2010. Springer-Verlag.
5. Sergei O. Kuznetsov. On the intractability of computing the Duquenne-Guigues base. *Journal of Universal Computer Science*, 10(8):927–933, 2004.
6. Sergei O. Kuznetsov and Sergei Obiedkov. Comparing performance of algorithms for generating concept lattices. *Journal of Experimental and Theoretical Artificial Intelligence*, 14:189–216, 2002.
7. Michael Luxenburger. Implications partielles dans un contexte. *Mathématiques et Sciences Humaines*, 113:35–55, 1991.
8. S. Obiedkov and V. Duquenne. Attribute-incremental construction of the canonical implication basis. *Annals of Mathematics and Artificial Intelligence*, 49(1-4):77–99, April 2007.
9. Petko Valtchev and Vincent Duquenne. On the merge of factor canonical bases. In *Proceedings of the 6th international conference on Formal concept analysis*, ICFCA'08, pages 182–198, Berlin, Heidelberg, 2008. Springer-Verlag.