Mining Formal Concepts using Implications between Items

Aimene Belfodil[1,2] ✉, Adnene Belfodil[1] ✉, and Mehdi Kaytoue[1,3]

[1] Univ Lyon, INSA Lyon, CNRS, LIRIS UMR 5205, F-69621, LYON, France
[2] Mobile Devices Ingénierie, 100 Avenue Stalingrad, 94800, Villejuif, France
[3] Infologic, 99 avenue de Lyon, 26500 Bourg-Lès-Valence, France
`firstname.surname@insa-lyon.fr`

Abstract. Formal Concept Analysis (FCA) provides a mathematical tool to analyze and discover concepts in Boolean datasets (i.e. Formal contexts). It does also provide a tool to analyze complex attributes by transforming them to Boolean ones (i.e. items) thanks to *conceptual scaling*. For instance, a numerical attribute whose values are $\{1, 2, 3\}$ can be transformed to the set of items $\{\leq 1, \leq 2, \leq 3, \geq 3, \geq 2, \geq 1\}$ thanks to interordinal scaling. Such transformations allow us to use standard algorithms like *Close-by-One* (`CbO`) to look for concepts in complex datasets by leveraging a closure operator. However, these standard algorithms do not use the relationships between attributes to enumerate the concepts as for example the fact that ≤ 1 implies ≤ 2 and so on. For such, they can perform additional closure computations which substantially degrade their performance. We propose in this paper a generic algorithm, named `CbOI` for *Close-by-One using Implications*, to enumerate concepts in a formal context using the inherent implications between items provided as an input. We show that using the implications between items can reduce significantly the number of closure computations and hence the time effort spent to enumerate the whole set of concepts.

1 Introduction

Formal Concept Analysis (FCA) [8, 25] provides a mathematical tool to analyze and discover concepts in Boolean datasets (i.e. Formal contexts). It does also provide a tool to analyze complex attributes by transforming them to Boolean ones (i.e. items) thanks to *conceptual scaling* [11]. For instance, a numerical attribute whose values are $\{1, 2, 3\}$ can be transformed to a set of items $\{\leq 1, \leq 2, \leq 3, \geq 3, \geq 2, \geq 1\}$ thanks to *interordinal scaling* (see Figure 1). Such transformations allow to use standard algorithms for enumerating all formal concepts in a formal context [5, 9, 17, 18] by leveraging a closure operator. However, these standard algorithms do not take advantage of the relationships between attributes to enumerate the concepts as for example the fact that "≤ 1 implies ≤ 2" and so on. For such, they perform additional closure computations which substantially degrade their performance. Some algorithms [2, 6, 14] have been proposed in the literature to handle some particular instances of contexts with implications. For instance, interordinal scaled contexts are directly linked to interval patterns as investigated by [14]. On the other hand, ordinal scaled contexts are linked to datasets augmented with a taxonomy (i.e hierarchy) between items (e.g. *lattices* are *posets*) [2, 6]. Yet, to the best of our knowledge, when a formal context is provided with an arbitrary set of implications (i.e. forming some directed graph between items), no generic algorithm is provided.

In this paper, we propose a generic algorithm, named `CbOI` for *Close-by-One using Implications*, to enumerate formal concepts using the inherent implications between items provided as an input. In other words, provided a pair (formal context, directed graph of implications between attributes), `CbOI` uses at its best the provided implications between items to enumerate exhaustively and non-redundantly formal concepts. The proposed algorithm relies on Boley et al. algorithm to enumerate closed sets in a strongly accessible set system [4]. In fact, we show that closed sets are upsets (i.e. upward closed) in an equivalent poset of the input directed graph. We use then the fact that the set of these upsets forms a strongly accessible set system since it is an anti-matroid [7]. Building on these notions, we elaborate algorithm `CbOI`.

This paper is organized as follow: Section 2 recalls basic definitions about binary relations and partially ordered sets as well as definitions from Formal Concept Analysis. Section 3 introduces Boley et al. Algorithm [4] and explain the drawbacks of not using the implications to enumerate concepts. Next, Section 4 presents the formalization of the problem as well as the newly proposed algorithm dubbed `CbOI`. Details about implementation[4] of `CbOI` as well as its experimental evaluation are presented and discussed in Section 5.

2 Preliminary Definitions

In this paper, given an arbitrary set E, $\wp(E)$ denotes the powerset of E (i.e. $\wp(E) = \{A \mid A \subseteq E\}$). Moreover, for any application $f : E \to F$, and for any subset $A \subseteq E$, $f[A]$ denotes the image of A by f (i.e. $f[A] = \{f(a) \mid a \in A\}$).

2.1 Binary Relations, Pre-Orders and Partial Orders

This section recalls basic definitions on binary relations, pre-ordered sets and partially ordered sets. More details can be found in [22]. Let E be an arbitrary set. A *binary relation* R over a set E is an arbitrary subset of $E \times E$. For any element (a, b) in R we denote aRb. A binary relation R is said to be: *(1) reflexive:* $(\forall a \in E)\ a\,R\,a$. *(2) transitive:* $(\forall a, b, c \in E)$ if $a\,R\,b$ and $b\,R\,c$ then $a\,R\,c$. *(3) symmetric:* $(\forall a, b \in E)$ if $a\,R\,b$ then $b\,R\,a$. and *(4) anti-symmetric:* $(\forall a, b \in E)$ if $a\,R\,b$ and $b\,R\,a$ then $a = b$.

A binary relation $\to$ on E is said to be a *pre-order* on E if it is *reflexive* and *transitive*. The pair $(E, \to)$ is said to be a *preordered-set* or a *proset* for short.

A binary relation $\leftrightarrow$ on E is said to be an *equivalence relation* on E if it is a *symmetric preorder* on E. The equivalence class of $a \in E$ the set of its equivalent

[4]**Source Code.** `https://github.com/BelfodilAimene/CbOImplications`

element $\{b \in E \mid b \leftrightarrow a\}$. The quotient set is the partition of E on equivalent classes and we denote it $E/\leftrightarrow$.

Partially Ordered Sets. A binary relation $\leq$ on E is said to be a *partial order* on E if it is an *anti-symmetric preorder* on E. The pair $(E, \leq)$ is said to be a *partially ordered set* or a *poset* for short.

A mapping $\sigma : E \to E$ is said to be a closure operator on a poset $(E, \leq)$ iff it is: *(1) extensive:* $(\forall a \in E)\ a \leq \sigma(a)$, *(2) monotonous:* $(\forall a, b \in E)$ if $a \leq b$ then $\sigma(a) \leq \sigma(b)$ and *(3) idempotent:* $(\forall a \in E)\ \sigma(\sigma(a)) = \sigma(a)$. The set $\sigma[E] = \{\sigma(e) \mid e \in E\}$ is then called the set of fixpoints of σ.

Up-sets and Up-closure. Given a poset $(E, \leq)$, a subset $S \subseteq E$ is said to be an up-set (or upper-ideal) *iff*: $(\forall x \in S, \forall y \in E)\ x \leq y \Rightarrow y \in S$. There is an operator $\uparrow$ on $\wp(E)$ that associates to any subset $S \subseteq E$, the smallest up-set enclosing it. It is given by: $\uparrow S = \{e \in E \mid (\exists s \in S)\ s \leq e\}$. Operator $\uparrow$ is called *up closure* and is a closure operator on $(\wp(E), \subseteq)$. The set of all up-sets, denoted $\mathcal{U}(E)$, is given by the set of all fix-points of $\uparrow$. It is closed under arbitrary intersection and arbitrary union (i.e. $(\mathcal{U}(E), \subseteq)$ does form a complete sublattice of $(\wp(E), \subseteq)$). Down-sets and down closure $\downarrow$ can be defined analogically. Please note that for $s \in S$, we denote $\uparrow s$ and $\downarrow s$ rather than $\uparrow \{s\}$ and $\downarrow \{s\}$. Sets $\uparrow s$ and $\downarrow s$ are called respectively principal filter and principal ideal. In fact, we have $\uparrow S = \bigcup_{s \in S} \uparrow s$ and $\downarrow S = \bigcup_{s \in S} \downarrow s$.

For a subset $S \subseteq E$, the set of minimal elements $min(S)$ and $max(S)$ are given respectively by the lower and the upper borders of S. Formally:

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

One important remark for a <u>finite</u> posets $(E, \leq)$ is that for any up-set $S \in \mathcal{U}(E)$ we have $S = \uparrow min(S)$. Moreover, $min(S)$ represents the smallest subset C in S (w.r.t. $\subseteq$) such that $S = \uparrow C$.

Last but not least, any finite poset $(E, \leq)$ can be represented by its Hasse Diagram [22]: the transitive reduction [1] of directed acyclic graph (dag) $(E, \leq)$. It does represent for each element $e \in E$ the set of its direct lower (resp. upper) neighbors $lowers(E)$ (resp. $uppers(E)$). For $e \in E$, we have:

$$lowers(e) = \{l \in E \mid e \text{ covers } l\} = max(\downarrow e \backslash \{e\})$$
$$uppers(e) = \{u \in E \mid u \text{ covers } e\} = min(\uparrow e \backslash \{e\})$$

For $e_1, e_2 \in E$, we say that e_2 *covers* e_1 iff $e_1 \neq e_2$, $e_1 \leq e_2$ and there is no element e lying strictly between e_1 and e_2: i.e. $(\forall e \in E)\ e_1 \leq e \leq e_2 \Rightarrow e = e_1$ or $e = e_2$.

2.2 Formal Concept Analysis

Formal Concept Analysis (FCA) was introduced in [25] as a mathematical framework to analyze and manipulate concepts in (binary) databases. *FCA* starts by a *formal context* [8]. A *(formal) context* is a triple $\mathbb{K} = (\mathcal{G}, \mathcal{M}, \mathcal{I})$ consisting of two sets $\mathcal{G}$ and $\mathcal{M}$ and an Incidence relation $\mathcal{I}$ between $\mathcal{G}$ and $\mathcal{M}$. Elements of

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

Table 1: A Formal Context.

$\mathcal{G}$ are called objects and elements of $\mathcal{M}$ are called attributes or items. In order to express that an object $g \in \mathcal{G}$ *has* an attribute $m \in \mathcal{M}$, we write $g\mathcal{I}m$ or $(g, m) \in \mathcal{I}$. A formal context $\mathbb{K}$ is said to be *finite* if $\mathcal{G}$ and $\mathcal{M}$ are both finites. Note that subsets of items $B \subseteq \mathcal{M}$ are called *patterns* or *itemsets*. Table 1 depicts a *formal context* $(\mathcal{G}, \mathcal{M}, \mathcal{I})$ where objects in $\mathcal{G}$ are places while attributes in $\mathcal{M}$ are place tags. The incidence relation $\mathcal{I}$ is represented by the crosses in the table and it does represent that a *place* is tagged by a *tag*. For instance, we have $place_3 \ \mathcal{I} \ Restaurant$ that can be read as: "*$place_3$ is tagged as a Restaurant*".

Two base operators, namely *extent* and *intent*, are defined on a formal context $\mathbb{K} = (\mathcal{G}, \mathcal{M}, \mathcal{I})$. The *extent operator*, denoted *ext*, associates to each itemset $B \subseteq \mathcal{M}$ the set of objects $g \in \mathcal{G}$ having all items in B. Formally, it is given by: $ext : \wp(\mathcal{M}) \to \wp(\mathcal{G}), B \mapsto \{g \in \mathcal{G} \mid (\forall m \in B) \ g\mathcal{I}m\}$. The set of all possible extents of a context $\mathbb{K}$ is denoted $\mathbb{K}_{ext}$ and is given by $\mathbb{K}_{ext} = ext[\wp(\mathcal{M})]$. Dually, the *intent operator*, denoted *int*, associates to each subset of objects $A \subseteq \mathcal{G}$ the set of items $m \in \mathcal{M}$ common to the objects in A. Formally, it is given by: $int : \wp(\mathcal{G}) \to \wp(\mathcal{M}), A \mapsto \{m \in \mathcal{M} \mid (\forall g \in A) \ g\mathcal{I}m\}$. The set of all possible intents of a context $\mathbb{K}$ is denoted $\mathbb{K}_{int}$ and is given by $\mathbb{K}_{int} = int[\wp(\mathcal{G})]$.

For ease of notations, for $g \in \mathcal{G}$ and $m \in \mathcal{M}$, we denote respectively $int(g)$ and $ext(m)$ rather than $int(\{g\})$ and $ext(\{m\})$. Note that, for $B \subseteq \mathcal{M}$ and $A \subseteq \mathcal{G}$, we have: $ext(B) = \bigcap_{m \in B} ext(m)$ and $int(A) = \bigcap_{g \in \mathcal{G}} int(g)$. A key theorem in FCA (Proposition 10 in [8]) is given below:

Theorem 1. *The pair of functions (ext, int) form a Galois connection between the powerset lattices $(\wp(\mathcal{G}), \subseteq)$ and $(\wp(\mathcal{M}), \subseteq)$. That is $ext \circ int$ and $int \circ ext$ are closure operators on $(\wp(\mathcal{G}), \subseteq)$ and $(\wp(\mathcal{M}), \subseteq)$ respectively. Hence, $\mathbb{K}_{ext}$ and $\mathbb{K}_{int}$ are* Moore Families *(i.e. closed under arbitrary intersection).*

This theorem allows to build what is called *concept lattice* $(\mathfrak{B}(\mathbb{K}), \leq)$ which is a complete lattice. Elements of $\mathfrak{B}(\mathbb{K})$ are called (formal) concepts and are of the form $(A, B) \in \wp(\mathcal{G}) \times \wp(\mathcal{M})$ such that $A = ext(B)$ and $B = int(A)$. In other words: $\mathfrak{B}(\mathbb{K}) = \{(ext(B), B) \mid B \in \mathbb{K}_{int}\} = \{(A, int(A)) \mid A \in \mathbb{K}_{ext}\}$. For two concepts (A_1, B_1) and (A_2, B_2) in $\mathfrak{B}(\mathbb{K})$, we say that (A_1, B_1) is a *subconcept* of (A_2, B_2) and we denote $(A_1, B_1) \leq (A_2, B_2)$ if $A_1 \subseteq A_2$ (or equivalently $B_2 \subseteq B_1$). From now on, elements of $\mathbb{K}_{int}$ are called *closed patterns* since they are the fixpoints of the closure operator $int \circ ext$.

As a matter of example, consider again the formal context depicted in Figure 2, we have $ext(\{hotel, restaurant\}) = \{place_2\}$, meaning that the only place that is both tagged *hotel* and *restaurant* is $place_2$. Dually $int(\{place_2, place_3\}) = \{Any, Restaurant\}$, that is what is common to $place_2$ and $place_3$ is that they are both tagged as *Any* and as *Restaurant*. Since $ext(\{Any, Restaurant\}) = \{place_2, place_3\}$, the pair $(\{place_2, place_3\}, \{Any, Restaurant\})$ is a concept.

$\mathcal{G}$	x	y
g_1	1	4
g_2	2	2
g_3	2	2
g_4	3	2

$\mathcal{G}$	$x \geq 1$	$x \geq 2$	$x \geq 3$	$x \leq 3$	$x \leq 2$	$x \leq 1$	$y \geq 2$	$y \geq 4$	$y \leq 4$	$y \leq 2$
g_1	×			×	×	×	×	×	×	
g_2	×	×		×	×		×		×	×
g_3	×	×		×	×		×	×	×	
g_4	×	×	×	×			×	×	×	

Fig. 1: (**left**) Numerical dataset with 2 numerical attributes. **(right)** A formal context that is the result of its *interordinal scaling*.

Another important notion in FCA tightly linked to formal concepts and closed sets are implications (see Definition 1). For example, we have in the context depicted in Table 1, that *Italian Restaurant* $\rightarrow$ *Restaurant*. Indeed, all *Italian Restaurant* are *Restaurant*.

Definition 1 (Item-Implications). *Let $A, B \subseteq \mathcal{M}$ be two itemsets. We say that A implies B and we denote $A \rightarrow B$ iff: $ext(A) \subseteq ext(B)$. In other words, if an object has all items in the set of attributes A then he has all items in the set of attributes B. For two items $a, b \in \mathcal{M}$, we call an implication $\{a\} \rightarrow \{b\}$* item-implication *and we denote it $a \rightarrow b$ for ease of notation.*

Handling Complex Data in Formal Concept Analysis While basic FCA provides a tool to analyze (formal) contexts (i.e. Boolean datasets), several techniques are proposed in FCA literature to handle more complex datasets like numerical ones. The most straightforward way, is called *(conceptual) scaling* [8,11]. It is the action of transforming a dataset to a formal context w.r.t. the nature of patterns we are looking for.

For instance, if we want to look for (closed) interval patterns in numerical datasets [14], we use *interordinal scaling*. Figure 1 depicts such a transformation. It can be shown that the set of closed patterns (i.e. intents) with a non-empty extent in the interordinal scaled context encoding a numerical dataset represents the set of non-empty extent closed interval patterns [14].

Another examples of datasets are those augmented by a taxonomy (i.e. hierarchy of items) [6]. For instance Figure 2 depicts such a dataset. In general, patterns we are looking for in such datasets are itemsets using elements of the taxonomy. Again, *ordinal scaling* can be used to handle such a type of datasets and pattern language using a formal context. The ordinal scaling of the dataset presented in Figure 2 is presented in Table 1. Please note that FCA offers other more sophisticated tools to handle complex data, we can cite among other techniques pattern structures [10,19] and pattern setups [3,21].

$\mathcal{G}$	*tags*
place$_1$	{Hotel}
place$_2$	{Chinese Restaurant, Hotel}
place$_3$	{Italian Restaurant}

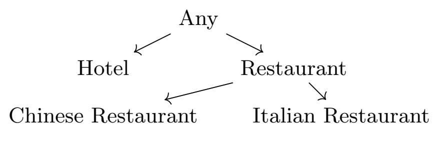


Fig. 2: (**left**) Point-of-interests annotated with tags and **(right)** a taxonomy of tags.

3 Formal Concepts Enumeration

The enumeration of formal concepts is an equivalent task to the enumeration of closed patterns in a formal context. Indeed, for a formal context $\mathbb{K}$, we have $\mathfrak{B}(\mathbb{K}) = \{(ext(C), C) \mid C \in \mathbb{K}_{int}\}$ where $\mathbb{K}_{int}$ is no more than the set of fixpoints of the closure operator $int \circ ext$. Many algorithms are proposed in the literature [5,9,17,18] (among others) to enumerate exhaustively and non-redundantly closed sets of some closure operator σ on $(\wp(E), \subseteq)$ with E an arbitrary finite set. A. Gély showed that many of those algorithms can be seen as an instance of a more general *divide-&-conquer* algorithm proposed in [12]. Later, Boley et al. [4] proved that such a generalization can be used to enumerate exhaustively and non-redundantly closed sets (i.e. fix-points) of a closure operator $\sigma : \mathcal{F} \to \mathcal{F}$ in a *finite strongly accessible set system* $(E, \mathcal{F})$ rather than only for the case $\mathcal{F} = \wp(E)$. We recall below some definitions about set systems.

A *set system* on an arbitrary finite set E is a pair $(E, \mathcal{F})$ where $\mathcal{F}$ is a set of subsets of E (i.e. $\mathcal{F} \subseteq \wp(E)$) called the set of *feasible sets*. For a set system $(E, \mathcal{F})$, we present below some properties:

- *[P0]* $\emptyset \in \mathcal{F}$.
- *[P1]* $\forall S \in \mathcal{F} \backslash \{\emptyset\}\ \exists e \in E$ s.t. $S \backslash \{e\} \in \mathcal{F}$.
- *[P2]* $\forall S, T \in \mathcal{F}$ s.t. $S \subsetneq T$ we have $(\exists e \in T \backslash S)\ S \cup \{e\} \in \mathcal{F}$.
- *[P3]* $\forall S, T \in \mathcal{F}$ s.t. $|S| < |T|$ we have $(\exists e \in T \backslash S)\ S \cup \{e\} \in \mathcal{F}$.
- *[P4]* $\forall S, T \in \mathcal{F}$ s.t. $T \not\subseteq S$ we have $(\exists e \in T \backslash S)\ S \cup \{e\} \in \mathcal{F}$.
- *[P5]* $\forall S, T \in \mathcal{F}$, $S \cup T \in \mathcal{F}$ (i.e. $\mathcal{F}$ is closed under set union).

Implications between properties are depicted in Figure 3. A set system $(E, \mathcal{F})$ is said to be *accessible* if it has property *[P1]* and *strongly accessible* [4] if it has both *[P1]* and *[P2]* properties. It is said to be a *greedoid* [15] if it has both *[P1]* and *[P3]* and an *anti-matroid* [7] if it has properties *[P0]* and *[P4]* (or equivalently *[P1]* and *[P5]*). Note that anti-matroids have the 6 aforementioned properties. *Anti-matroids* are *greedoids* and *greedoids* are *strongly accessibles.*

Given a finite strongly accessible set system $(E, \mathcal{F})$ and a closure operator $\sigma : \mathcal{F} \to \mathcal{F}$. Boley et al. [4] showed that the algorithm, dubbed *Divide & Conquer Closed Set Listing* (`D&C` for short), enumerates exhaustively and non-redundantly fixpoints of σ (i.e. $\sigma[\mathcal{F}]$). Algorithm 1 starts from the smallest element $\sigma(\emptyset)$ in $\sigma[\mathcal{F}]$ (Line 8) then enumerates in depth-first fashion concepts in $\sigma[\mathcal{F}]$ by performing closure computations (Line 3) then checking, thanks to canonicity test (Line 4), if the closed set is already generated making `D&C` non-redundant.

Concepts enumeration and eventual problems. Algorithm `D&C` can be instantiated easily for instance to enumerate closed patterns (or concepts) of a formal context $\mathbb{K} = (\mathcal{G}, \mathcal{M}, \mathcal{I})$ in a top-down fashion. Indeed, the considered

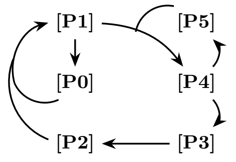


Fig. 3: Implications between set system properties (in *finite* set systems). Note that *[P0] and [P2]* together imply *[P1]*; and *[P1] and [P5]* together imply *[P4]*.

Algorithm 1: D&C (Divide & Conquer Closed Set Listing) Algorithm

```
Input: Finite strongly accessible set system $(E, \mathcal{F})$ and
       a closure operator $\sigma$ on $(\mathcal{F}, \subseteq)$.
Output: Elements of $\sigma[\mathcal{F}]$
1 procedure D&C$(C, B)$
2     for $e \in E \backslash (B \cup C)$ s.t. $C \cup \{e\} \in \mathcal{F}$ do
3         $C_{new} \leftarrow \sigma(C \cup \{e\})$      // Compute the new closed element $C_{new}$
4         if $C_{new} \cap B = \emptyset$ then
5             D&C$(C_{new}, B)$
6         $B \leftarrow B \cup \{e\}$                // Update the set of banned elements $B$
7     Print$(C)$                        // Output the closed element $C \in \sigma[\mathcal{F}]$
8 D&C$(\sigma(\emptyset), \emptyset)$
```

set system can be the general one $(\mathcal{M}, \wp(\mathcal{M}))$ and the closure operator is $int \circ ext$. Rather than outputting only closed patterns $C \in \mathbb{K}_{int}$ (Line 7), one can print the concept $(ext(C), C)$. Such an algorithm is in fact Close-by-One (CbO) [17,18]. Algorithm CbO has a delay time complexity[5] of $\mathcal{O}(|\mathcal{G}||\mathcal{M}|^2)$, a total time complexity of $\mathcal{O}(|\mathcal{G}||\mathcal{M}|^2|\mathbb{K}_{int}|)$. Please note that the time complexity of the closure computation is given by $\mathcal{O}(|\mathcal{G}||\mathcal{M}|)$.

However, such an enumeration does not use the inherent implications between items. For instance, let us reconsider the *interordinal scaled context* presented in Figure 1, we present below some steps of execution of CbO:

1. **Begins:** $C_0 = \{x \geq 1, x \leq 3, y \geq 2, y \leq 4\}$ and $B = \emptyset$.
2. **Add item $\mathbf{x \geq 2}$ to C_0:** $C_{new} = \{x \geq 1, \mathbf{x \geq 2}, x \leq 3, y \geq 2, y \leq 4\}$ at Line 3. The canonicity test does not fail since $B = \emptyset$ and algorithm continues by enumerating all subconcepts of $(ext(C_{new}), C_{new})$. Further, at line 6, we have $B = \{\mathbf{x \geq 2}\}$.
3. **Add item $\mathbf{x \geq 3}$ to C_0:** $C_{new} = \{x \geq 1, \underline{x \geq 2}, \mathbf{x \geq 3}, x \leq 3, y \geq 2, y \geq 4, y \leq 4\}$ at Line 3. Since $B = \{\underline{x \geq 2}\}$, canonicity test fails since $C_{new} \cap B = \{\underline{x \geq 2}\}$. The enumeration continues until the end.

The problem shown beforehand after adding $x \geq 3$ is the fact that there was a useless closure computation that led to a certain failure. One could avoid this closure computation if the inherent implication $x \geq 3 \rightarrow x \geq 2$ is used. Indeed, since $x \geq 3 \rightarrow x \geq 2$, any closed itemset containing $x \geq 3$ contains $x \geq 2$. This shows that one can avoid some closure computations if implications are used properly, or in other words, avoid visiting some non valid closed itemsets. Moreover, such implications are sometime known from the user since they are *inherent* to the attributes and not derived from the incidence relation of the context. This is the case of interordinal and ordinal scaled datasets for example. While some state-of-the-art algorithms try to use this knowledge (i.e. implications between some items of the context) in some particular datasets as

[5] **Delay time:** maximum time between two outputs, between the beginning and the first output and between the last output and the ending of an enum. algo. (cf. [13])

it is the case of numerical datasets (interordinal scaled contexts) [14] or datasets augmented with a taxonomy of items (ordinal scaled contexts) [2,6]; no general algorithm has been proposed to enumerate concepts in a context while taking benefit from an arbitrary provided set of item-implications (cf. Definition 1).

4 Using Item-Implications to Enumerate Concepts

In this section, we start by formalizing the user inputs, that is: the formal context and the implications between items provided by the user. Next, we show how to transform this pair of inputs to an equivalent pair with a poset between (sets of) items. We show then that the closed patterns of the input context are, up to the aforementioned transformation, in the anti-matroid of up-sets of the poset (cf. Proposition 2 and 4). The proposed algorithm is then a straightforward implementation of `D&C` since anti-matroids are strongly accessibles.

Let $\mathbb{K} = (\mathcal{G}, \mathcal{M}, \mathcal{I})$ be a finite formal context.

Definition 2. *The* item-implication basis *associated to a context* $\mathbb{K}$*, denoted* $\rightarrow$*, is given below:*

$$\rightarrow := \{(a, b) \in \mathcal{M} \times \mathcal{M} \mid a \rightarrow b\} = \{(a, b) \in \mathcal{M} \times \mathcal{M} \mid ext(a) \subseteq ext(b)\}$$

The definition given beforehand regroups all item-implications existing in the context. Moreover, pair $(\mathcal{M}, \rightarrow)$ forms a pre-ordered set. We model now the item-implication basis known/provided by the user. We can say informally that such a set of implications are those that are inherent to the attributes (not necessarily derived) from the incidence relation.

Definition 3. *A* valid Item-Implication basis *for* $\mathbb{K}$ *is any* sub relation $\mathfrak{I}$ *of* $\rightarrow$.

User Inputs. The input is then a pair $(\mathbb{K}, \mathfrak{I})$ where $\mathbb{K}$ is a finite formal context and $\mathfrak{I}$ is a valid item-implication basis for $\mathbb{K}$. Figure 4 (left) depicts an example of the input pair $(\mathbb{K}, \mathfrak{I})$. It is clear that $\mathfrak{I}$ provides a partial information about a pre-order. Relation $\mathfrak{I}$ can be augmented to a sub pre-order $\rightarrow_{\mathfrak{I}}$ of $\rightarrow$ thanks to *reflexive closure* (i.e. adding (m, m) to $\mathfrak{I}$ for all $m \in \mathcal{M}$) and *transitive closure* (i.e. the smallest transitive relation containing $\mathfrak{I}$). Thus we will be dealing from now on with an equivalent pair $(\mathbb{K}, \rightarrow_{\mathfrak{I}})$ where $(\mathcal{M}, \rightarrow_{\mathfrak{I}})$ is a pre-ordered set.

4.1 Building a Partial Order

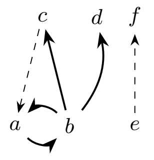
Pre-order $\rightarrow_{\mathfrak{I}}$ could contain some cycles (i.e. not anti-symmetric). One can define an equivalence relation $\leftrightarrow_{\mathfrak{I}}$ on $\mathcal{M}$ such that $(\forall a, b \in \mathcal{M})\ a \leftrightarrow_{\mathfrak{I}} b$ iff $a \rightarrow_{\mathfrak{I}} b$ and $b \rightarrow_{\mathfrak{I}} a$. Please note that if $a \leftrightarrow_{\mathfrak{I}} b$ then $ext(a) = ext(b)$. With $\mathcal{M}' = \mathcal{M}/\leftrightarrow_{\mathfrak{I}}$ the quotient set of $\mathcal{M}$ on $\leftrightarrow_{\mathfrak{I}}$ and the following relation $\leq_{\mathfrak{I}}$:

$$(\forall S_1, S_2 \in \mathcal{M}')\ S_1 \leq_{\mathfrak{I}} S_2 \text{ iff } (\exists a \in S_1, \exists b \in S_2)\ a \rightarrow_{\mathfrak{I}} b$$

One can show that $(\mathcal{M}', \leq_{\mathfrak{I}})$ does form a partially ordered set (poset). Accordingly, context $\mathbb{K} = (\mathcal{G}, \mathcal{M}, \mathcal{I})$ is also transformed to $\mathbb{K}' = (\mathcal{G}, \mathcal{M}', \mathcal{I}')$ where:

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

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



$\mathcal{G}$	$\{\mathbf{a},\mathbf{b}\}$	$\{c\}$	$\{d\}$	$\{e\}$	$\{f\}$
g_1	×	×	×		
g_2	×	×	×	×	
g_3				×	×
g_4			×		

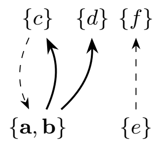


Fig. 4: (**left**) Input context $\mathbb{K} = (\mathcal{G}, \mathcal{M}, \mathcal{I})$, the input-item implications $\mathfrak{I}$ (continuous arrows) and the item-implications of the context that are valid but not provided (dashed arrows). **(right)** The result of transformation to the context $\mathbb{K}' = (\mathcal{G}, \mathcal{M}', \mathcal{I}')$ and the Hasse diagram of the poset $(\mathcal{M}', \leq_{\mathfrak{I}})$ (continuous arrows).

Figure 4 gives an example of such a transformation from $\mathbb{K}$ to $\mathbb{K}'$ and from $(\mathcal{M}, \rightarrow_{\mathfrak{I}})$ to $(\mathcal{M}, \leq_{\mathfrak{I}})$. Note that this transformation is a *partial column-clarification* (see [8]) in the sense that it does concern only the item-implication basis $\rightarrow_{\mathfrak{I}}$ provided by the user since we want to use only the *user inputs*. If the total item-implication basis $\rightarrow$ associated with the context is used, the before-hand transformation will be equivalent to a column clarification. Proposition 1 provides that looking for concepts (or extents) in $\mathbb{K}$ is equivalent to look for concepts in the partially clarified context $\mathbb{K}'$. As such, from now on, we will consider the pair $(\mathbb{K}, \leq_{\mathfrak{I}})$ such that $(\mathcal{M}, \leq_{\mathfrak{I}})$ is a partial order. If not so, the context and the item-implication basis are transformed as shown beforehand (cf. Figure 4).

Proposition 1. *We have* $\mathbb{K}_{ext} = \mathbb{K}'_{ext}$.

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

- $\mathbb{K}'_{ext} \subseteq \mathbb{K}_{ext}$: Let $A \in \mathbb{K}'_{ext}$, that is $\exists \mathbb{S} \subseteq \mathcal{M}'$ such that $ext_{\mathbb{K}'}(\mathbb{S}) = A$, that is: $A = \bigcap_{S \in \mathbb{S}} ext_{\mathbb{K}'}(\{S\})$. However, it is clear that: $ext_{\mathbb{K}'}(\{S\}) = \{g \in \mathcal{G} \mid g\,\mathcal{I}'\,S\} = \{g \in \mathcal{G} \mid (\forall m \in S)\, g\,\mathcal{I}\,m\} = ext_{\mathbb{K}}(S)$. Hence, $A = \bigcap_{S \in \mathbb{S}} ext_{\mathbb{K}}(S)$. In other words, $A \in \mathbb{K}_{ext}$ since it is the intersection of some elements in $\mathbb{K}_{ext}$.
- $\mathbb{K}_{ext} \subseteq \mathbb{K}'_{ext}$: Let $A \in \mathbb{K}_{ext}$, that is: $\exists B \subseteq \mathcal{M}$ such that $ext_{\mathbb{K}}(B) = A$, that is: $A = \bigcap_{m \in B} ext_{\mathbb{K}}(\{m\})$. For $m \in B$, let $S_m \in \mathcal{M}'$ be the unique set containing m. We have $ext_{\mathbb{K}'}(\{S_m\}) = ext_{\mathbb{K}}(\{m\})$. Hence, $A = \bigcap_{m \in B} ext_{\mathbb{K}'}(\{S_m\})$; that is $A \in \mathbb{K}'_{ext}$ since it is an intersection of some elements in $\mathbb{K}'_{ext}$. □

4.2 Closed Patterns Are Up-Sets

Consider now the obtained pair $(\mathbb{K}, \leq_{\mathfrak{I}})$ where $\mathbb{K} = (\mathcal{G}, \mathcal{M}, \mathcal{I})$ is a context and $(\mathcal{M}, \leq_{\mathfrak{I}})$ is a poset s.t. $\forall a, b \in \mathcal{M}$, we have if $a \leq_{\mathfrak{I}} b$ then $ext(a) \subseteq ext(b)$.

Lemma 1. $\forall C \subseteq \mathcal{M} : \uparrow C \subseteq int \circ ext(C)$

Proof. Let $m \in \uparrow C$, thus $\exists c \in C$ such that $c \rightarrow_{\mathfrak{I}} m$ or in other words, $ext(c) \subseteq ext(m)$. Since int is order-reversing ((ext, int) is a Galois connection (see Theorem 1)), $int \circ ext(m) \subseteq int \circ ext(c)$. Since $int \circ ext$ is monotonous then $int \circ ext(c) \subseteq int \circ ext(C)$. Hence, $int \circ ext(m) \subseteq int \circ ext(C)$. By extensivity of $int \circ ext$ we conclude that $m \in int \circ ext(C)$; that is $\uparrow C \subseteq int \circ ext(C)$. □

A straightforward corollary of Lemma 1 is given below:

Proposition 2. *Closed patterns are up-sets on* $(\mathcal{M}, \leq_{\mathfrak{I}})$ *that is* $\mathbb{K}_{int} \subseteq \mathcal{U}(\mathcal{M})$ *where* $\mathcal{U}(\mathcal{M}) = \{S \subseteq \mathcal{M} \mid \uparrow S = S\}$ *is the set of upsets on* $(\mathcal{M}, \leq_{\mathfrak{I}})$.

Proof. Let $C \in \mathbb{K}_{int}$, we have $C \subseteq \uparrow C$ by extensivity of $\uparrow$. In the other hand, we have $\uparrow C \subseteq int \circ ext(C) = C$ according to Proposition 1 and by using the impotence of $int \circ ext$. Hence, $C = \uparrow C$ or in other words $C \in \mathcal{U}(\mathcal{M})$. □

Proposition 2 shows that all closed patterns are up-sets. It should be noticed that if the provided item-implication basis $\mathfrak{I}$ is equal to the complete item-implication basis $\rightarrow$ associated to context (see Definition 2), then all principal filters in poset $(\mathcal{M}, \leq_{\mathfrak{I}})$ are in $\mathbb{K}_{int}$ (i.e. $(\forall m \in \mathcal{M})\ int \circ ext(m) = \uparrow m$).

We conclude that: rather than enumerating elements of $\wp(\mathcal{M})$ to look for closed itemsets, as for instance Close-by-One does, one can consider only elements of $\mathcal{U}(\mathcal{M})$. For that, one can use Algorithm `D&C` (see Algorithm 1) to enumerate concepts of $\mathbb{K}$ if the set system $(\mathcal{M}, \mathcal{U}(\mathcal{M}))$ is strongly accessible which is fortunately the case according to Proposition 4. We state in Proposition 3 a characterization of the upper neighbors of $S \in \mathcal{U}(\mathcal{M})$ in the poset $(\mathcal{U}(\mathcal{M}), \subseteq)$. This proposition is crucial to build the algorithm and to prove Proposition 4.

Proposition 3. *Let* $(\mathcal{M}, \leq)$ *be a* <u>*finite*</u> *poset, we have:*

$$(\forall S \in \mathcal{U}(\mathcal{M}), \forall a \in \mathcal{M} \backslash S)\ \ S \cup \{a\} \in \mathcal{U}(\mathcal{M}) \textbf{\textit{ iff }} a \in max(\mathcal{M} \backslash S)$$

Proof. Let $S \in \mathcal{U}(\mathcal{M})$ and let $a \in \mathcal{M} \backslash S$. We show below both implications:
– ($\Leftarrow$) Let $a \in max(\mathcal{M} \backslash S)$, that is: $(\uparrow a) \cap (\mathcal{M} \backslash S) = \{a\}$. Hence, $\uparrow a \subseteq S \cup \{a\}$. We have $\uparrow (S \cup \{a\}) = \uparrow S \cup \uparrow \{a\} = S \cup \uparrow \{a\}$. Since, $\uparrow \{a\} \subseteq S \cup \{a\}$, then $S \cup \uparrow \{a\} \subseteq S \cup \{a\}$. By extensivity of $\uparrow$ we have $S \cup \{a\} \subseteq S \cup \uparrow \{a\}$. Thus, $\uparrow (S \cup \{a\}) = S \cup \{a\}$ or in other words $S \cup \{a\} \in \mathcal{U}(\mathcal{M})$.
– ($\Rightarrow$) Let $a \notin max(\mathcal{M} \backslash S)$, that is $\exists b \in \mathcal{M} \backslash S$ such that $a \leq b$ and $a \neq b$. Hence, $b \in \uparrow (S \cup \{a\})$ (since $b \in \uparrow a$) but in the same time $b \notin S \cup \{a\}$. In other words $\uparrow (S \cup \{a\}) \neq S \cup \{a\}$. Hence, $S \cup \{a\} \notin \mathcal{U}(\mathcal{M})$. □

Figure 5 depicts for $C \in \mathcal{U}(\mathcal{M})$ the set $max(\mathcal{M} \backslash C)$. Hence, the only direct upper neighbors of $C = \{\mathbf{a}, \mathbf{c}, \mathbf{d}\}$ in $(\mathcal{U}(\mathcal{M}), \subseteq)$ according to Proposition 3 are $\{\mathbf{a}, \mathbf{c}, \mathbf{d}, \underline{b}\}$, $\{\mathbf{a}, \mathbf{c}, \mathbf{d}, \underline{e}\}$ and $\{\mathbf{a}, \mathbf{c}, \mathbf{d}, \underline{f}\}$ since $max(\mathcal{M} \backslash C) = \{\underline{b}, \underline{e}, \underline{f}\}$.

Proposition 4. *For* $(\mathcal{M}, \leq)$ *a* <u>*finite*</u> *poset,* $(\mathcal{M}, \mathcal{U}(\mathcal{M}))$ *is an anti-matroid [7].*

Proof. We have $\emptyset \in \mathcal{U}(\mathcal{M})$. It remains to show now that $(\mathcal{M}, \mathcal{U}(\mathcal{M}))$ has property $[P4]$ (see Section 3). Let be two upsets $A, C \in \mathcal{U}(\mathcal{M})$ s.t. $C \nsubseteq A$, we need to show that: $\exists a \in C \backslash A$ s.t. $A \cup \{a\} \in \mathcal{U}(\mathcal{M})$.

Let us show before that $max(C \backslash A) \subseteq max(\mathcal{M} \backslash A)$. Let $e \in max(C \backslash A)$, hence $e \in \mathcal{M} \backslash A$. Let $f \in \mathcal{M}$ s.t. $e \leq f$ and $e \neq f$. In one hand, since $e \in C$ we have $f \in C$ since $C \in \mathcal{U}(\mathcal{M})$. In the other hand, since $e \in max(C \backslash A)$ then $f \notin C \backslash A$. Therefore, $f \in A$ or in other words $f \notin \mathcal{M} \backslash A$. We conclude that $e \in max(\mathcal{M} \backslash A)$. Thus $max(C \backslash A) \subseteq max(\mathcal{M} \backslash A)$.

According to Proposition 3, $\forall a \in max(\mathcal{M} \backslash A)$, we have $A \cup \{a\} \in \mathcal{U}(\mathcal{M})$. Moreover, we have $max(C \backslash A) \neq \emptyset$ since $\mathcal{M}$ is finite and $C \backslash A \neq \emptyset$ (since $C \nsubseteq A$). Since $\emptyset \neq max(C \backslash A) \subseteq max(\mathcal{M} \backslash A)$ then $\exists a \in C \backslash A$ s.t. $A \cup \{a\} \in \mathcal{U}(\mathcal{M})$. □

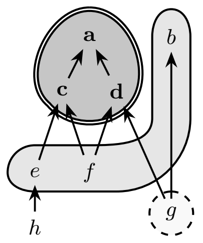


Fig. 5: Poset $(\mathcal{M}, \leq_{\mathfrak{I}})$ with $\mathcal{M} = \{a, b, c, d, e, f, g, h\}$. The set *contoured with two lines* $C = \{\mathbf{a}, \mathbf{c}, \mathbf{d}\} \in \mathcal{U}(\mathcal{M})$ is an upset and $min(C) = \{c, d\}$. The set *contoured with one line* $\texttt{addables}(C) = max(\mathcal{M} \backslash C) = \{b, e, f\}$ regroups addable items. Indeed, all upper neighbors of items b, e and f are in C (note that item b has no upper-neighbor). The set *contoured with one dashed line* $\texttt{potential\_addables}(C) = \{g\}$ contains potential addable items. Indeed, $uppers(g) = \{b, \mathbf{d}\}$ contains at least one element in C.

4.3 Close-by-One using Implications (CbOI) Algorithm

We have shown in Proposition 2 that, for a $(\mathbb{K}, \leq_{\mathfrak{I}})$ with $\mathbb{K} = (\mathcal{G}, \mathcal{M}, \mathcal{I})$ s.t. $(\mathcal{M}, \leq_{\mathfrak{I}})$ is a poset, all closed patterns in $\mathbb{K}_{int}$ are upsets in $\mathcal{U}(\mathcal{M})$. Since $(\mathcal{M}, \mathcal{U}(\mathcal{M}))$ is strongly accessible (Proposition 4), we can use Algorithm 1 (D&C) to enumerate concepts in $\mathbb{K}$ using closure operator $int \circ ext$ on $(\mathcal{M}, \mathcal{U}(\mathcal{M}))$. Algorithm 2 dubbed *Close-by-One using Implications (CbOI for short)* is then a straightforward implementation of Algorithm 1 where somehow only line 2 is modified according to Proposition 3: For $C \in \mathbb{K}_{int}$, the set $\texttt{addables}(C)$ denotes the set of items to add to build the next closed itemsets. It is given by:

$$\texttt{addables}(C) = max(\mathcal{M} \backslash C) = \{a \in \mathcal{M} \backslash C \mid uppers(a) \subseteq C\}$$

In the next section, we show that some optimizations can be made to compute and maintain efficiently the set of addable items for each generated closed sets (Line 2). Moreover, one can partially compute the closure (Line 3-4) in order to perform canonicity test (Line 5).

Algorithm 2: Algorithm *Close-by-One using Implications* (CbOI)

```
Input: Pair $(\mathbb{K}, \leq_{\mathfrak{I}})$ with $\mathbb{K} = (\mathcal{G}, \mathcal{M}, \mathcal{I})$ a formal context and
       $(\mathcal{M}, \leq_{\mathfrak{I}})$ is a poset s.t. if $m_1 \leq_{\mathfrak{I}} m_2$ then $ext(m_1) \subseteq ext(m_2)$.
Output: Elements of $\mathfrak{B}(\mathbb{K})$
1 procedure CbOI($A, C, B$)
2     for $m \in$ addables$(C) \backslash B$ do
3         $A_{new} \leftarrow A \cap ext(m)$          // Compute the new extent $A_{new}$
4         $C_{new} \leftarrow int(A_{new})$          // Compute the new intent $C_{new}$
5         if $C_{new} \cap B = \emptyset$ then
6             CbOI($A_{new}, C_{new}, B$)
7         $B \leftarrow B \cup \{m\}$        // Update the set of banned attributes $B$
8     Print($A$, $C$)                  // Output the extent $A$ and the intent $C$
9 CbOI($\mathcal{G}$, $int(\mathcal{G})$, $\emptyset$)
```

5 Empirical Evaluation and Technical Details

We start by explaining some technical details around CbOI implementation provided in https://github.com/BelfodilAimene/CbOImplications.

5.1 Implementation Details

Computing the Partial Order from the Input Item-Implications. As explained in Section 4.1, the user input is a pair $(\mathbb{K}^{(0)}, \mathfrak{I}^{(0)})$ where context $\mathbb{K}^{(0)} = (\mathcal{G}, \mathcal{M}^{(0)}, \mathcal{I}^{(0)})$ a finite context and $\mathfrak{I} \subseteq \mathcal{M}^{(0)} \times \mathcal{M}^{(0)}$ such that for all $(m_1, m_2) \in \mathfrak{I}$ we have $ext(m_1) \subseteq ext(m_2)$ (i.e. valid item-implication basis). One can model this provided item-implication basis as a directed graph $(\mathcal{M}^{(0)}, \mathfrak{I}^{(0)})$. The aim at the beginning, is to compute the associated partial order and the partial scaling of the context w.r.t. to the item implication basis. To do so we start by computing the set of strongly connected components $\mathcal{M}$ on the directed graph $(\mathcal{M}^{(0)}, \mathfrak{I}^{(0)})$. This can be done for instance using Tarjan's Algorithm [24] whose complexity is $\mathcal{O}(|\mathcal{M}^{(0)}| + |\mathfrak{I}^{(0)}|)$. Once done, we can build the associated directed acyclic graph (DAG) $(\mathcal{M}, \mathfrak{I}^{(1)})$ where there is an arc (S_1, S_2) in $\mathfrak{I}^{(1)}$ iff $\exists m_1 \in S_1$ and $m_2 \in S_2$ such that $(m_1, m_2) \in \mathfrak{I}^{(0)}$. Such an operation is called *graph condensation*. Once $\mathcal{M}$ computed, the context $\mathbb{K}^{(0)}$ is transformed to an equivalent context $\mathbb{K} = (\mathcal{G}, \mathcal{M}, \mathcal{I})$ (i.e. $\mathbb{K}_{ext} = \mathbb{K}^{(0)}_{ext}$ according to Proposition 1).

The DAG $(\mathcal{M}, \mathfrak{I}^{(1)})$ represents the partial order $\leq_{\mathfrak{I}}$ (i.e. a reflexive and transitive closure of $\mathfrak{I}$ creates $\leq_{\mathfrak{I}}$). A more usual and efficient way to store $\leq_{\mathfrak{I}}$ is to compute the *transitive reduction* of $(\mathcal{M}, \mathfrak{I}^{(1)})$ [1] to obtain the Hasse Diagram of $\leq_{\mathfrak{I}}$. We obtain then $(\mathcal{M}, \mathfrak{I})$ where we store for each $m \in \mathcal{M}$ both sets of its direct lower and upper neighbors (i.e. $lowers(m)$ and $uppers(m)$) w.r.t. $(\mathcal{M}, \leq_{\mathfrak{I}})$.

Computing and Maintaining Addable Items. Now that the partial order $\leq_{\mathfrak{I}}$ is encoded by the list of lower and upper neighbors for each item $m \in \mathcal{M}$ (i.e. the Hasse Diagram). One solution is that at each step of the algorithm, for a closed itemset C, we computed the set $\texttt{addables}(C) = \{a \in \mathcal{M} \backslash C \mid uppers(a) \subseteq C\}$ whose complexity is $\mathcal{O}(\mathcal{G}^2)$. This computation of addable items could lessen the performances of the implementation. To address this drawback, we propose to keep for each generated itemset $C \in \mathcal{U}(\mathcal{M})$, the set of its addable items $\texttt{addables}(C)$, the set of potential addable items $\texttt{potential\_addables}(C)$ and its minimal elements $min(C)$ (only if we want to output them). An item $p \in \mathcal{M}$ is said to be *potentially addable* if $p \in \mathcal{M} \backslash (C \cup \texttt{addables}(C))$ and it has at least one element of its direct upper neighbors $uppers(p)$ in C. Formally: $\texttt{potential\_addables}(C) = (\bigcup_{c \in min(C)} lowers(c)) \backslash \texttt{addables}(C)$. Figure 5 gives an example about addable and potential addable items for a closed itemset C.

These three aforementioned sets of addable items, potential addable items and minimal items can be maintained incrementally as follow. Given an up-set $C \in \mathcal{U}(\mathcal{M})$ and an item $a \in \texttt{addables}(C)$, the following steps are performed to compute the three sets associated to the up-set $C \cup \{a\}$:

1. $min(C \cup \{a\}) := (min(C) \backslash uppers(a)) \cup \{a\}$.
2. $\texttt{potential\_addables}(C \cup \{a\}) := \texttt{potential\_addables}(C) \cup lowers(a)$
3. Initialize addable items $\texttt{addables}(C \cup \{a\})$ by $\texttt{addables}(C) \backslash \{a\}$.
4. For each item p in the computed $\texttt{potential\_addables}(C \cup \{a\})$, if we have $uppers(p) \subseteq min(C \cup \{a\})$ then remove it from $\texttt{potential\_addables}(C \cup \{a\})$ and add it to $\texttt{addables}(C \cup \{a\})$. This can be further optimized by maintaining for each potentially addable item the number of direct upper

neighbors that are not already in C. Whenever, element a is add, we subtract 1 from the values associated to elements in $lowers(a)$. Once a potentially addable element sees its value become 0, he is considered as addable item and no longer potentially addable.

Going back to Figure 5, adding the addable item $\underline{b}$ updates the different sets as follow: $C_{new} = \{\mathbf{a}, \mathbf{c}, \mathbf{d}, \underline{b}\}$, $min(C_{new}) = \{\mathbf{c}, \mathbf{d}, \underline{b}\}$, $\texttt{addables}(C_{new}) = \{e, f, \underline{\underline{g}}\}$ and $\texttt{potential\_addables}(C_{new}) = \emptyset$.

Computing Next Closure and Performing Canonicity Test. Line 3-5 in Algorithm 2 are dedicated to closure computation of the newly generated set and checking if such a closed pattern is already generated. Some optimizations can be made here. For instance, vertical representation of the context (i.e. keeping for each item, its extent) can be held in memory in order to compute efficiently the new pattern extent (Line 3). For closure computation (Line 4) and canonicity test (Line 5), one can use the optimizations explained below:

1. We have a canonicity test fail (i.e. $int \circ ext(C \cup \{m\}) \cap B \neq \emptyset$) iff $\exists b \in B \cap \texttt{addables}(C)$ such that $ext(C \cup \{m\}) \subseteq ext(b)$. Hence, we do not need to compute the closure $int \circ ext(C \cup \{m\})$ to perform the canonicity test. Note that to ensure a fair comparison between `CbOI` and `CbO`, this same optimization has been used for `CbO` implementation.
2. To maintain both sets of addable and potential addable items as explained beforehand, closure computation is computed incrementally by adding item per item until there is no addable item a s.t. $ext(C \cup \{m\}) \subseteq ext(a)$.

Outputting Minimal Elements. If the item-implications in $\leq_{\mathfrak{I}}$ are well-known by the user, one should output only minimal element of a closed pattern C w.r.t. $\leq_{\mathfrak{I}}$ (i.e. $min(C)$) since C contains some redundant information [6, 14].

5.2 Experimental Evaluation

Experiment Settings. Experiments were conducted in a machine with an Intel Core i7-7700HQ 2.80GHz CPU and *7.7 GiB* memory space and the implementation was done using Python 2.7.12. Table 2 reports the benchmark input contexts and their associated item-implications basis. *Europarl*[6] and *Yelp*[7] are datasets augmented with a taxonomy. Hence, their corresponding contexts $\mathbb{K}_1$ and $\mathbb{K}_2$ are obtained via an ordinal scaling and their associated implication basis are derived from the hierarchy of items induced by the provided taxonomy. *Basketball*[8], *Airport*[8] and *Iris*[9] are numerical datasets. Analogously, their corresponding contexts $\mathbb{K}_3$, $\mathbb{K}_4$ and $\mathbb{K}_5$ are the result of an interordinal scaling and the associated implication basis are constituted with two chains of implications per attribute (i.e. if the domain of the numerical attribute is $\{1, 2, 3\}$ then the item implications basis associated to the inter-ordinal scaling is given by $\leq 1 \rightarrow \leq 2 \rightarrow \leq 3$

[6]EPD8 (last accessed on 04 Octobre 2018): `http://parltrack.euwiki.org/`
[7]Yelp (last accessed on 25 April 2017): `www.yelp.com/dataset/challenge`
[8]Bilkent repository: `http://funapp.cs.bilkent.edu.tr/`
[9]UCI repository: `https://archive.ics.uci.edu/ml/index.php`

	context $\mathbb{K} = (\mathcal{G}, \mathcal{M}, \mathcal{I})$				**implications** $\mathfrak{I}$		
$(\mathbb{K}, \mathfrak{I})$	Name	$\lvert\mathcal{G}\rvert$	$\lvert\mathcal{M}\rvert$	$\lvert\mathbb{K}_{int}\rvert$	$\lvert \rightarrow^*_{\mathfrak{I}} \rvert$	$\lvert \rightarrow^* \rvert$	density
$(\mathbb{K}_1, \mathfrak{I}_1)$	Europarl	4 742	357	1 307	709	1 034	68.57%
$(\mathbb{K}_2, \mathfrak{I}_2)$	Yelp	127 162	1 174	63 300	1 514	2 111	71.72%
$(\mathbb{K}_3, \mathfrak{I}_3)$	Basketball	40	272	272 223	4 716	13 724	34.36%
$(\mathbb{K}_4, \mathfrak{I}_4)$	Iris	150	246	6 516 292	3 964	11 704	33.87%
$(\mathbb{K}_5, \mathfrak{I}_5)$	Airport	135	1 348	82 467 125	90 182	313 432	28.77%
$(\mathbb{K}_6, \mathfrak{I}_6)$	Mushrooms	8 124	119	238 710	0	949	0.00%

Table 2: Benchmark Inputs and their characteristics: the number of objects $|\mathcal{G}|$, the number of attributes $|\mathcal{M}|$, the size of the concept lattice $|\mathbb{K}_{int}|$ of the context $\mathbb{K}$, the number of strict (irreflexive) item-implications $| \rightarrow^*_{\mathfrak{I}} |$ in the pre-order associated to the corresponding input implication basis, the number of strict item-implications in the context $| \rightarrow^* |$ (see Definition 2) and the density given by $| \rightarrow^*_{\mathfrak{I}} |/| \rightarrow^* |$.

and $\geq 3 \rightarrow \geq 2 \rightarrow \geq 1$). *Mushroom*[9] features only nominal attributes. Hence, its associated context $\mathbb{K}_6$ represents the result of nominal scaling of all attributes. Note that the set of implications $\mathfrak{I}_6$ is empty, yet there are some implications between items that are context-dependent (i.e. $\rightarrow$ is not empty).

Evaluation Results. Table 3 reports the number of closures and the performance of `CbO` and `CbOI` on the different benchmark inputs. For each benchmark context $\mathbb{K}_i$, we run both algorithms on the provided implication basis (i.e. input $(\mathbb{K}_i, \mathfrak{I}_i)$) as well as on the total one that is associated to the context (i.e. $(\mathbb{K}_i, \rightarrow)$). For a fair comparison, we report the context load/preparation time into the memory for `CbO` as well as the load/preparation time of the pair (context, implication basis) for `CbOI`. When $\rightarrow$ implication basis is used, the load time includes the time spent to compute it.

It is clear that the number of closures performed by `CbOI` is much less than the ones performed by `CbO` in all tests excepts when no implications are provided. This corresponds to the case $(\mathbb{K}_6, \mathfrak{I}_6)$ where the number of closures performed by `CbOI` is supposed to be equivalent to the number of closures performed by `CbO` if the same order of choice of items to add is followed.

Concerning the execution time, it is clear that the load time for `CbO` is lesser than the load/compute time for `CbOI` since `CbOI` does load and prepare additionally the item-implication basis. However, even if `CbOI` has this drawback, one can see that the enumeration time is much faster than `CbO` (i.e. up to $15\times$ faster for input $(\mathbb{K}_3, \mathfrak{I}_3)$ or even more for input $(\mathbb{K}_4, \mathfrak{I}_4)$). This compensates the overhead induced by the implication-basis load time in `CbOI`. One could notice that `CbO` perform better than `CbOI` when no implication is provided as it is the case in test $(\mathbb{K}_6, \mathfrak{I}_6)$. This is due to the fact that `CbOI` manages more structures than `CbO` during enumeration. It is worth noting that even if the implication basis is computed then used to enumerate concepts (see tests $(\mathbb{K}_i, \rightarrow)$), `CbOI` performs faster than `CbO` (up to $6\times$ faster for input $(\mathbb{K}_3, \mathfrak{I}_3)$). Still, we can observe that `CbOI` is less efficient when the underlying implication basis is huge (case $(\mathbb{K}_3, \rightarrow)$, $(\mathbb{K}_4, \rightarrow)$ and $(\mathbb{K}_5, \rightarrow)$). This can be explained by the fact that `CbOI` spends more time to handle a huge and complex system of item-implications but the gain obtained from these base of implications does not compensate this effort.

	`CbO`				`CbOI`			
$(\mathbb{K}, \mathfrak{I})$	nb closure	load (ms)	enum (ms)	total (ms)	nb closure	load (ms)	enum (ms)	total (ms)
$(\mathbb{K}_1, \mathfrak{I}_1)$	185 418	**86**	184	270	**17 020**	220	**48**	**268**
$(\mathbb{K}_1, \rightarrow)$					**13 409**	191	**46**	**237**
$(\mathbb{K}_2, \mathfrak{I}_2)$	24 437 659	**8 715**	30 360	39 075	**3 317 590**	18 084	**16 107**	**34 191**
$(\mathbb{K}_2, \rightarrow)$					**2 974 130**	19 976	**15 030**	**35 006**
$(\mathbb{K}_3, \mathfrak{I}_3)$	13 340 233	**3**	57 286	57 289	**703 999**	20	**3 628**	**3 648**
$(\mathbb{K}_3, \rightarrow)$					**445 735**	39	**9 114**	**9 153**
$(\mathbb{K}_4, \mathfrak{I}_4)$	170 615 166	**10**	709 517	709 527	**9 618 493**	26	**77 586**	**77 612**
$(\mathbb{K}_4, \rightarrow)$					**8 383 741**	73	**141 016**	**141 089**
$(\mathbb{K}_5, \mathfrak{I}_5)$	NA	**53**	$> 12h$	$> 12h$	**122 717 962**	268	**1 496 175**	**1 496 443**
$(\mathbb{K}_5, \rightarrow)$					**106 409 230**	1 400	**8 221 648**	**8 223 048**
$(\mathbb{K}_6, \mathfrak{I}_6)$	**4 363 487**	**155**	**13 800**	**13 955**	4 363 511	184	15 985	16 169
$(\mathbb{K}_6, \rightarrow)$	4 363 487	**155**	13 800	13 955	**1 338 245**	244	**10 003**	**10 247**

Table 3: `CbO` and `CbOI` performance comparison on the benchmark inputs

6 Conclusion

In this paper, we have investigated how to incorporate and leverage the inherent implications between items in some given context so as to enumerate more efficiently its formal concepts. Experimental studies demonstrated that the proposed algorithm dubbed `CbOI` for *Close-by-One using Implications* is far more efficient than its concurrent `CbO` in most configurations. Indeed, many aspects of the devised algorithm can be considerably improved. For instance, including `FCbO` optimizations [16] during the enumeration process can significantly reduce the number of falsely generated closed patterns. Moreover, the load/preparation time of `CbOI` can be more enhanced by, for example, computing more efficiently the transitive reduction of the implication basis [20]. Another important remark is that the same notions here can be used to look for minimal generators. In fact, Kaytoue et al. [14] showed that there is no one-to-one correspondence between the minimal generators in the interval pattern structure and the minimal generators in the interordinal scaled contexts conversely to closed interval patterns. However, if we consider only up-sets w.r.t. the poset induced by the implications, there is a one-to-one correspondence between interval patterns and up-sets. Hence, minimal generators for interval patterns can be mined efficiently by algorithm `DeFMe` [23] since it considers strongly accessible set systems.

Aknowledgement. This work has been partially supported by the project *ContentCheck* **ANR-15-CE23-0025** funded by the French National Research Agency and the Association Nationale Recherche Technologie (**ANRt**) French program. The authors would like to thank the reviewers for their valuable remarks. They also warmly thank Anes Bendimerad for interesting discussions.

References

1. Aho, A.V., Garey, M.R., Ullman, J.D.: The transitive reduction of a directed graph. SIAM J. Comput. **1**(2), 131–137 (1972)

2. Belfodil, A., Cazalens, S., Lamarre, P., Plantevit, M.: Flash points: Discovering exceptional pairwise behaviors in vote or rating data. In: ECML/PKDD (2). pp. 442–458 (2017)
3. Belfodil, A., Kuznetsov, S.O., Kaytoue, M.: Pattern setups and their completions. In: CLA. pp. 243–253 (2018)
4. Boley, M., Horváth, T., Poigné, A., Wrobel, S.: Listing closed sets of strongly accessible set systems with applications to data mining. Theor. Comput. Sci. **411**(3), 691–700 (2010)
5. Bordat, J.P.: Calcul pratique du treillis de galois d'une correspondance. Mathématiques et Sciences humaines **96**, 31–47 (1986)
6. Cellier, P., Ferré, S., Ridoux, O., Ducassé, M.: An algorithm to find frequent concepts of a formal context with taxonomy. In: CLA. pp. 226–231 (2006)
7. Dietrich, B.L.: Matroids and antimatroidsa survey. Discrete Mathematics **78**(3), 223–237 (1989)
8. Ganter, B., Wille, R.: Formal Concept Analysis. Springer (1999)
9. Ganter, B.: Two basic algorithms in concept analysis. Technical report, Technische Hoschule Darmstadt (1984)
10. Ganter, B., Kuznetsov, S.O.: Pattern structures and their projections. In: ICCS. pp. 129–142 (2001)
11. Ganter, B., Wille, R.: Conceptual scaling. In: Applications of combinatorics and graph theory to the biological and social sciences, pp. 139–167. Springer (1989)
12. Gély, A.: A generic algorithm for generating closed sets of a binary relation. In: ICFCA. pp. 223–234 (2005)
13. Johnson, D.S., Papadimitriou, C.H., Yannakakis, M.: On generating all maximal independent sets. Inf. Process. Lett. **27**(3), 119–123 (1988)
14. Kaytoue, M., Kuznetsov, S.O., Napoli, A.: Revisiting Numerical Pattern Mining with Formal Concept Analysis. In: IJCAI. pp. 1342–1347 (2011)
15. Korte, B., Lovász, L.: Mathematical structures underlying greedy algorithms. In: International Conference on Fundamentals of Computation Theory. pp. 205–209 (1981)
16. Krajca, P., Outrata, J., Vychodil, V.: Advances in algorithms based on cbo. In: CLA. pp. 325–337 (2010)
17. Kuznetsov, S.O.: A Fast Algorithm for Computing All Intersections of Objects in a Finite Semi-lattice. Nauchno-Tekhnicheskaya Informatsiya **ser. 2**(1), 17–20 (1993)
18. Kuznetsov, S.O.: Learning of simple conceptual graphs from positive and negative examples. In: PKDD. pp. 384–391 (1999)
19. Kuznetsov, S.O.: Pattern structures for analyzing complex data. In: RSFDGrC 2009. pp. 33–44 (2009)
20. Le Gall, F.: Powers of tensors and fast matrix multiplication. In: Proceedings of the 39th international symposium on symbolic and algebraic computation. pp. 296–303. ACM (2014)
21. Lumpe, L., Schmidt, S.E.: Pattern structures and their morphisms. In: CLA. vol. 1466, pp. 171–179 (2015)
22. Roman, S.: Lattices and Ordered Sets. Springer New York (2008)
23. Soulet, A., Rioult, F.: Efficiently depth-first minimal pattern mining. In: PAKDD 2014. pp. 28–39. Springer (2014)
24. Tarjan, R.E.: Depth-first search and linear graph algorithms. SIAM J. Comput. **1**(2), 146–160 (1972)
25. Wille, R.: Restructuring lattice theory: an approach based on hierarchies of concepts. In: Ordered Sets. pp. 445–470 (1982)