*Fundamenta Informaticae 115 (2012) 395–417*
*DOI 10.3233/FI-2012-661*
*IOS Press*



# Computing Formal Concepts by Attribute Sorting

**Petr Krajca, Jan Outrata*, Vilem Vychodil†‡**
*DAMOL: Data Analysis and Modeling Laboratory*
*Department of Computer Science*
*Palacky University, Olomouc, Czech Republic*
*vychodil@acm.org; jan.outrata@upol.cz; petr.krajca@binghamton.edu*

**Abstract.** We present a novel approach to compute formal concepts of formal context. In terms of operations with Boolean matrices, the presented algorithm computes all maximal rectangles of the input Boolean matrix which are full of 1s. The algorithm combines basic ideas of previous approaches with our recent observations on the influence of attribute permutations and attribute sorting on the number of formal concepts which are computed multiple times. As a result, we present algorithm which computes formal concepts by successive context reduction and attribute sorting. We prove its soundness, discuss its complexity and efficiency, and show that it outperforms other algorithms from the CbO family in terms of substantially lower numbers of formal concepts which are computed multiple times.

## 1. Introduction and Problem Setting

Formal concept analysis (FCA) is a method of relational data analysis proposed by R. Wille [27] in early 80's. Since its inception, there has been an extensive theoretical research which has lead to many order-theoretical results, see [7] for a survey. Another, maybe equally important fact is that the results have been directly applied to various fields of data analysis including analysis in software engineering [25, 26], web information retrieval [11], and market-basket analysis [29]. Examples of FCA applications can be found in [4, 7].

*Jan Outrata was supported by grant no. P202/10/P360 of the Czech Science Foundation.
†Petr Krajca and Vilem Vychodil were supported by grant no. P103/10/1056 of the Czech Science Foundation.
‡Address for correspondence: Department of Computer Science, Palacky University, 17. listopadu 12, CZ–77146 Olomouc, Czech Republic

In its basic setting, FCA deals with object-attribute relational data which can be seen as a data table with rows corresponding to objects, columns corresponding to attributes (features), and table entries being 1 or 0, indicating whether objects have or do not have corresponding attributes. Formally, such data tables can be seen as binary relations between a set of objects and a set of attributes. The aim of FCA is to extract from such input data useful information about interesting object-attribute biclusters and attribute dependencies which are present in data. The outputs of FCA are used either directly or for preprocessing purposes. In the first case, extracted object-attribute clusters (so-called formal concepts) are ordered by a subconcept-superconcept hierarchy and can be presented to users by a line diagram of clusters (diagram of so-called concept lattice). The users can then navigate though the hierarchy to find clusters, identified by sets of objects and attributes that are covered by the clusters, which represent interesting and/or useful information for them. For instance, in an object-attribute database of cars and their features, users can find clusters like "affordable and safe cars", "four-wheel drive SUVs", etc., which they may find interesting. Note that the interpretation of a cluster as a concept having its extent (objects that fall under the concept) and its intent (attributes that fall under the concept) which is used in FCA is inspired by a traditional understanding of concept which goes back to Port-Royal logic [5, 18].

If FCA is used for preprocessing, the extracted clusters (formal concepts) are not used by users directly. Instead, they are used as input for other data mining methods. For instance, the seminal paper [24] showed that the formal concepts can be used to find non-redundant association rules, cf. also [29]. Recently, it has been shown in [3] that formal concepts can be used to find optimal factorization of Boolean matrices. In fact, it can be shown that they correspond to optimal solutions of the discrete basis problem discussed by Miettinen et al. [21].

In either case, the basic computational problem of FCA is to compute, given an input formal context (an object-attribute data table), the set of all formal concepts (the object-attribute clusters present in the input data). In the past, there have been proposed various algorithms for solving this task, see [17] for a survey and comparison. Among the best-known algorithms are CbO [14, 15, 16] proposed by Kuznetsov, Ganter's NextClosure [6, 7], and Lindig's UpperNeighbor [19] algorithms. There is an important family of algorithms which includes CbO, NextClosure, the algorithm proposed by Norris [22], and other algorithms such as PCbO [12], FCbO [13, 23], and InClose [2]. We call this family a *CbO family* because all algorithms in the family can be seen as modifications or refinements of CbO. For instance, NextClosure can be seen an iterative version of CbO, PCbO is a parallel variant of CbO, FCbO is a refinement of CbO which uses a new canonicity test, etc. In a broader sense, the CbO family of algorithms can be seen as an example of a family of algorithms for listing combinatorial structures [8].

A common issue that all algorithms for FCA have to care about is to prevent processing (e.g., storing or listing) the same formal concept multiple times. There are several approaches to cope with the problem. The CbO family algorithms use canonicity tests which are generally very cheap to perform. The basic idea is the following. Formal concepts are supposed to be computed in a predefined order. If the order is not preserved in a certain branch of computation (i.e., a newly computed formal concept does not pass the canonicity test during the computation), the branch is no longer considered. As a consequence, the canonicity test ensures that even if a formal concept is computed several times, it is processed (e.g., stored or listed) exactly once.

Although conceptually similar, algorithms from the CbO family differ in their efficiency. One of the most important factors is just the efficiency of the underlying canonicity tests. For instance, FCbO

uses a canonicity test which is more efficient than that of the original CbO. In practice, the numbers of formal concepts which are computed multiple times by FCbO is considerably smaller than the numbers corresponding to CbO [13, 23]. Another efficiency issue which is related to canonicity tests is the order in which attributes are processed by algorithms of the CbO family. In general, an important feature of algorithms for FCA is whether their performance depends on the order of objects and attributes in the input formal context. From this point of view, we shall call an algorithm (*permutation*) *resistant* whenever all isomorphic copies (in the usual sense) of the input formal context require the same number of elementary computation steps in order to compute all concepts. For our purposes, an elementary computation step shall be represented by computation of a single formal concept. One can easily see that, e.g., Lindig's UpperNeighbor algorithm [19] is resistant. In other words, if we rearrange rows and columns in the input data table, the algorithm uses the exact same number of steps to compute all formal concepts. On the other hand, algorithms from the CbO family are not resistant [13] and thus considering different orders of attributes can reduce the number of concepts that are computed multiple times, thus improving the efficiency.

The present paper is partly motivated by our observations from [13] where we have investigated the impact of using different orders of attributes for algorithms from the CbO family. One of the results presented in [13] says that if attributes of formal context are sorted in the ascending order according to their supports, i.e., the numbers of objects having the attributes, then the canonicity test of both CbO and FCbO always succeeds for all attribute concepts (concepts generated by a single attribute) provided that all attributes are distinct (i.e., all columns of the input data table are pairwise distinct). Furthermore, our empirical experiments have shown an interesting tendency that while processing formal contexts with attributes sorted in the aforementioned order, canonicity tests tend to fail less frequently than in the case of contexts containing inversions (with respect to the aforementioned order). In addition, with increasing number of inversions in a data table, the average number of computed closures grows. This seems to be a general tendency which has been experimentally observed in [13].

In the present paper, we elaborate on the ideas of attribute sorting. Motivated by the results of attribute sorting presented in [13], we introduce a method for attribute sorting and context reduction which is performed after obtaining a new formal concept. Unlike the approach in [13], where attribute sorting was just a means of data preprocessing and was used for each input data exactly once (before the computation which is then done by standard CbO or FCbO), we utilize attribute sorting during the computation several times which results in a conceptually new algorithm. The idea of dynamic reordering of attributes appeared in algorithm CHARM [28] for computing closed itemsets. In the paper, we describe the algorithm, prove its soundness, and investigate its complexity and further efficiency issues related to efficiency of its canonicity test. As we shall see in further sections, in terms of the numbers of concepts computed multiple times, the proposed algorithm outperforms CbO by an order of magnitude. The improvement is apparent especially in the case of large real data sets [9].

The paper is organized as follows. Section 2 contains brief preliminaries from FCA. Section 3 introduces operations with formal contexts which are used to describe the algorithm. Section 4 introduces the algorithm. Section 5 contains a detailed running example of the algorithm. Section 6 contains proof of soundness of the algorithm. Finally, Section 7 is devoted to complexity and efficiency issues of the algorithm and contains performance comparison with other algorithm from the CbO family.

## 2. Preliminaries from FCA

In this section we recall basic notions of FCA. More details can be found in monographs [7] and [4]. Let $X$ and $Y$ denote finite sets of objects and attributes, respectively. A formal context is a triple $\mathbb{K} = \langle X, Y, I \rangle$ where $I \subseteq X \times Y$, i.e. $I$ is a binary relation between $X$ and $Y$. The fact $\langle x, y \rangle \in I$ is interpreted so that "object $x$ has attribute $y$". Note that $\mathbb{K}$ obviously corresponds to a two-dimensional data table with rows corresponding to objects from $X$, columns corresponding to attributes from $Y$, and table entries being 1 and 0 indicating whether $\langle x, y \rangle \in I$ or $\langle x, y \rangle \notin I$. Thus, formal contexts can be seen as Boolean matrices.

Given $\mathbb{K} = \langle X, Y, I \rangle$, we introduce a pair of concept-forming operators [7] ${}^{\uparrow_{\mathbb{K}}} : 2^X \to 2^Y$ and ${}^{\downarrow_{\mathbb{K}}} : 2^Y \to 2^X$ defined, for each $A \subseteq X$ and $B \subseteq Y$, by $A^{\uparrow_{\mathbb{K}}} = \{y \in Y \mid \text{for each } x \in A : \langle x, y \rangle \in I\}$ and $B^{\downarrow_{\mathbb{K}}} = \{x \in X \mid \text{for each } y \in B : \langle x, y \rangle \in I\}$, respectively. If there is no danger of confusion, we omit $\mathbb{K}$ and write just ${}^{\uparrow}$ and ${}^{\downarrow}$ instead of ${}^{\uparrow_{\mathbb{K}}}$ and ${}^{\downarrow_{\mathbb{K}}}$, respectively. The cardinality of $\{y\}^{\downarrow_{\mathbb{K}}}$ is called the support of $y \in Y$. By a formal concept in $\mathbb{K}$ with extent $A$ and intent $B$ we mean any pair $\langle A, B \rangle \in 2^X \times 2^Y$ such that $A^{\uparrow_{\mathbb{K}}} = B$ and $B^{\downarrow_{\mathbb{K}}} = A$. Thus, formal concepts are fixed points of the concept-forming operators. Intuitively, each formal concept $\langle A, B \rangle$ represents a bicluster in $\mathbb{K}$ which consists of objects $A$ that fall under the concept and attributes $B$ that fall under the concept. Since $A^{\uparrow_{\mathbb{K}}} = B$ and $B^{\downarrow_{\mathbb{K}}} = A$, $A$ is a set of objects having all attributes from $B$ and $B$ is a set of attributes shared by all objects from $A$. Let us stress that formal concepts can be seen as maximal Boolean submatrices in the following sense: any $\langle A, B \rangle \in 2^X \times 2^Y$ such that $A \times B \subseteq I$ can be called a Boolean submatrix of $\mathbb{K}$ (which is full of 1s). Moreover, a Boolean submatrix $\langle A, B \rangle$ of $\mathbb{K}$ is a maximal one if, for each Boolean submatrix $\langle A', B' \rangle$ of $\mathbb{K}$ such that $A \times B \subseteq A' \times B'$, we have $A = A'$ and $B = B'$. We have that $\langle A, B \rangle \in 2^X \times 2^Y$ is a maximal Boolean submatrix of $\mathbb{K}$ (which is full of 1s) iff $A^{\uparrow_{\mathbb{K}}} = B$ and $B^{\downarrow_{\mathbb{K}}} = A$. Hence, maximal Boolean submatrices full of 1s are exactly the formal concepts.

The set of all formal concepts in $\mathbb{K} = \langle X, Y, I \rangle$ will be denoted by $\mathcal{B}(X, Y, I)$. Recall that $\mathcal{B}(X, Y, I)$ endowed by a concept ordering $\leq$ forms a complete lattice, called a concept lattice, whose structure is described by the Basic Theorem of FCA [7, 27].

## 3. Clarification and Attribute Sorting

In this section, we introduce basic operations with contexts that are used to describe the proposed algorithm for computing formal concepts. One of the distinguishing features of the algorithm is that during the computation, it transforms an initial formal context into other contexts by taking subsets of objects and by grouping several attributes together. In addition to that, groups of attributes are sorted according to their support and equipped with an additional numerical flag indicating whether a group of attributes is allowed to be present in intents of formal concepts computed in next stages (a precise meaning of the flag will be described later). These operations on contexts play a crucial role and will be described in this section. We begin with particular representation of formal contexts.

### 3.1. Input Formal Contexts and $R$-contexts

Here we describe the basic form of formal concepts which are used during the computation. As in case of any algorithm for computing formal concepts, the input for our algorithm is a formal context

$\mathbb{K} = \langle X, Y, I \rangle$. In order to keep information about groups of attributes, we use particular contexts, called $R$-contexts, to represent input data. A formal definition follows.

**Definition 3.1.** Given a formal context $\mathbb{K} = \langle X, Y, I \rangle$, a triple $\mathbb{K}^\sharp = \langle X^\sharp, Y^\sharp, I^\sharp \rangle$ is called an *$R$-context* (*derived from* $\mathbb{K}$) if the following conditions are satisfied:

(i) $X^\sharp \subseteq X$;

(ii) $Y^\sharp \subseteq \mathbb{N}_0 \times 2^Y$ such that for any $\langle n_1, B_1 \rangle \in Y^\sharp$ and $\langle n_2, B_2 \rangle \in Y^\sharp$ we have either that (a) $n_1 = n_2$ and $B_1 = B_2 \neq \emptyset$ or (b) $B_1 \neq \emptyset$, $B_2 \neq \emptyset$, and $B_1 \cap B_2 = \emptyset$;

(iii) for any $x \in X^\sharp$ and $\langle n, B \rangle \in Y^\sharp$: $\langle x, y_1 \rangle \in I$ iff $\langle x, y_2 \rangle \in I$ holds true for all $y_1, y_2 \in B$;

(iv) $I^\sharp = \{\langle x, \langle n, B \rangle\rangle \in X^\sharp \times Y^\sharp \mid \langle x, y \rangle \in I \text{ for all } y \in B\}$.

In addition, $\mathbb{K}^\sharp = \langle X^\sharp, Y^\sharp, I^\sharp \rangle$ is called an *initial $R$-context* (*derived from* $\mathbb{K}$) if $X^\sharp = X$, $Y^\sharp = \{\langle 0, \{y\}\rangle \mid y \in Y\}$, and $I^\sharp = \{\langle x, \langle 0, \{y\}\rangle\rangle \in X^\sharp \times Y^\sharp \mid \langle x, y \rangle \in I\}$. ■

We can immediately observe basic properties of $R$-contexts:

**Remark 3.2.** (a) Each $R$-context is a formal context. Notice that due to (iv), $\langle x, \langle n, B \rangle\rangle \in I^\sharp$ iff $x \in B^{\downarrow_{\mathbb{K}}}$ for $x \in X^\sharp$ and $\langle n, B \rangle \in Y^\sharp$. Moreover, taking into account (iii) and (iv), it follows that for any $\langle x, \langle n, B \rangle\rangle \in X^\sharp \times Y^\sharp$, $\langle x, \langle n, B \rangle\rangle \in I^\sharp$ iff there is $y \in B$ such that $\langle x, y \rangle \in I$ in which case $\langle x, y \rangle \in I$ is true for all $y \in B$ because of (iii). Note that each attribute $\langle n, B \rangle \in Y^\sharp$ has two parts: a numerical flag $n$ (explained later) and a subset $B \subseteq Y$ of original attributes. Using (ii), we get that $B \neq \emptyset$. In addition to that, distinct attributes from $Y^\sharp$ have associated pairwise disjoint nonempty subsets of original attributes.

(b) Note that attributes in $R$-context $\mathbb{K}^\sharp = \langle X^\sharp, Y^\sharp, I^\sharp \rangle$ have natural interpretation as sets of attributes from the original context which are indistinguishable in $\mathbb{K}$ provided we restrict ourselves only to objects from $X^\sharp$. Indeed, this is a basic consequence of Definition 3.1 (iii).

(c) An initial $R$-context derived from $\mathbb{K}$ is an $R$-context. Indeed, (i) and (ii) are obvious since attributes of an initial $R$-context are all of the form $\langle 0, \{y\}\rangle$. It is immediate that (iii) and (iv) of Definition 3.1 are satisfied as well. Obviously, an initial $R$-context $\mathbb{K}^\sharp$ derived from $\mathbb{K}$ is isomorphic to $\mathbb{K}$ in the usual sense. In other words, $\mathbb{K}^\sharp$ is exactly the same as $\mathbb{K}$ up to the names of attributes.

From now on, we describe further operations with contexts in terms of $R$-contexts instead of the original input contexts. By this we do not impose any restriction since an initial $R$-context derived from $\mathbb{K}$ has the same concepts up to different names of attributes, see Remark 3.2 (c).

**Example 3.3.** As an example, we consider a formal context $\mathbb{K}$ with objects $X = \{a, \ldots, f\}$ and attributes $Y = \{0, \ldots, 7\}$. The context (left) and an $R$-context derived from $\mathbb{K}$ (right) are depicted in Table 1. Notice that the original attributes 1 and 4 are distinguishable in $\mathbb{K}$ by object $c$. On the other hand, they are indistinguisbahle on $\{b, d, e, f\}$, hence the attribute $\langle 0, \{1, 4\}\rangle$ in $Y^\sharp$ is correct and satisfies the requirement given by Definition 3.1 (iii). Also, note that all attributes in $\mathbb{K}^\sharp$ except for $\langle 1, \{2\}\rangle$ are given zero flags.

**Remark 3.4.** Note that $\mathbb{K}^\sharp$ which results from $\mathbb{K}$ is fully given by the sets $X^\sharp$ and $Y^\sharp$ of objects and attributes, respectively. The binary relation $I^\sharp$ can be determined from the original $I$, see Remark 3.2 (a). Thus, a concise computer representation of $\mathbb{K}^\sharp$ can consist of a list of objects and attributes, respectively,

Table 1. Formal context $\mathbb{K}$ (left) and an $R$-context derived from $\mathbb{K}$ (right)

| $\mathbb{K}$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| $a$ | | × | × | | × | | | |
| $b$ | × | | × | × | | × | | × |
| $c$ | | | | × | × | | | × |
| $d$ | × | × | × | | × | × | × | × |
| $e$ | × | × | | | × | × | | |
| $f$ | × | | × | × | | × | | × |

| $\mathbb{K}^\sharp$ | $\langle 0,\{1,4\}\rangle$ | $\langle 1,\{2\}\rangle$ | $\langle 0,\{3\}\rangle$ | $\langle 0,\{6\}\rangle$ | $\langle 0,\{7\}\rangle$ |
|---|---|---|---|---|---|
| $b$ | | × | × | | × |
| $d$ | × | × | | × | × |
| $e$ | × | | | | |
| $f$ | | × | × | | × |

omitting the expensive operation of copying a part of the data representation of $I$ which can be kept in computer memory only once.

We conclude this subsection by showing that the concept-forming operators induced by $R$-contexts have a close relationship to concept-forming operators of original contexts. In order to keep concise notation, we first introduce the following abbreviation. For any $D \subseteq Y^\sharp$, we define

$$\lfloor D \rfloor = \bigcup\{B \subseteq Y \mid \langle n, B\rangle \in D\}. \tag{1}$$

Using this notation, we have:

**Lemma 3.5.** Let $\mathbb{K}^\sharp = \langle X^\sharp, Y^\sharp, I^\sharp\rangle$ be an $R$-context derived from $\mathbb{K}$. Then, for any $C \subseteq X^\sharp$ and $D \subseteq Y^\sharp$,

$$\lfloor C^{\uparrow_{\mathbb{K}^\sharp}} \rfloor = C^{\uparrow_{\mathbb{K}}} \cap \lfloor Y^\sharp \rfloor, \tag{2}$$

$$X^\sharp \cap \lfloor D \rfloor^{\downarrow_{\mathbb{K}}} = D^{\downarrow_{\mathbb{K}^\sharp}}. \tag{3}$$

**Proof:**
Both equalities can be proved using basic properties of $R$-contexts.

"(2)": Let $y \in \lfloor C^{\uparrow_{\mathbb{K}^\sharp}} \rfloor$. Therefore, there is $\langle n, B\rangle \in C^{\uparrow_{\mathbb{K}^\sharp}} \subseteq Y^\sharp$ such that $y \in B$. Hence, $y \in \lfloor Y^\sharp \rfloor$. Moreover, $\langle n, B\rangle \in C^{\uparrow_{\mathbb{K}^\sharp}}$ yields that for each $x \in C$, $\langle x, \langle n, B\rangle\rangle \in I^\sharp$. Due to Definition 3.1 (iv), the latter means that for each $y \in B$ and $x \in C$, we have $\langle x, y\rangle \in I$. Therefore, $B \subseteq C^{\uparrow_{\mathbb{K}}}$, i.e., $y \in C^{\uparrow_{\mathbb{K}}}$, showing $\lfloor C^{\uparrow_{\mathbb{K}^\sharp}} \rfloor \subseteq C^{\uparrow_{\mathbb{K}}} \cap \lfloor Y^\sharp \rfloor$. Conversely, take $y \in C^{\uparrow_{\mathbb{K}}} \cap \lfloor Y^\sharp \rfloor$. Then, for each $x \in C$, we have that $\langle x, y\rangle \in I$. Since $y \in \lfloor Y^\sharp \rfloor$, there is $\langle n, B\rangle \in Y^\sharp$ such that $y \in B$. Since $C \subseteq X^\sharp$, using Definition 3.1 (iii) and the previous fact, we get that for each $y \in B$ and $x \in C$, $\langle x, y\rangle \in I$. Thus, for each $x \in C$, $\langle x, \langle n, B\rangle\rangle \in I^\sharp$, meaning $\langle n, B\rangle \in C^{\uparrow_{\mathbb{K}^\sharp}}$ and thus $y \in B \subseteq \lfloor C^{\uparrow_{\mathbb{K}^\sharp}} \rfloor$.

"(3)": Consider $x \in X^\sharp \cap \lfloor D \rfloor^{\downarrow_{\mathbb{K}}}$. Therefore, for each $y \in \lfloor D \rfloor$, $\langle x, y\rangle \in I$. In particular, for any $B \subseteq \lfloor D \rfloor$ such that $\langle n, B\rangle \in D$, we have $\langle x, y\rangle \in I$ for all $y \in B$, i.e., $\langle x, \langle n, B\rangle\rangle \in I^\sharp$ since $x \in X^\sharp$. Moreover, $\langle n, B\rangle \in D$ has been taken arbitrarily, which means that $x \in D^{\downarrow_{\mathbb{K}^\sharp}}$. Conversely, let $x \in D^{\downarrow_{\mathbb{K}^\sharp}}$. By definition, we have $\langle x, \langle n, B\rangle\rangle \in I^\sharp$ for all $\langle n, B\rangle \in D$. Therefore, $\langle x, y\rangle \in I$ for all $y \in B$ such that $\langle n, B\rangle \in D$, meaning that $\langle x, y\rangle \in I$ is true for all $y \in \lfloor D \rfloor$. Therefore, $x \in \lfloor D \rfloor^{\downarrow_{\mathbb{K}}}$. The fact that $x \in X^\sharp$ is trivial. □

### 3.2. Clarification

Each $R$-context can be transformed into a new $R$-context with possibly smaller sets of attributes by a process of clarification. Recall from [7] that a formal context $\mathbb{K} = \langle X, Y, I \rangle$ is called clarified if for any $y_1, y_2 \in Y$ it follows that $\{y_1\}^{\downarrow} = \{y_2\}^{\downarrow}$ implies $y_1 = y_2$ and dually for any couple of objects. In other words, a clarified context in sense of [7] is a formal context where all columns in the corresponding object-attribute data table are distinct and dually for rows. It is a well known fact that taking a clarified formal context (with duplicate rows and columns removed) instead of the original one we get a possibly smaller context whose concept lattice is isomorphic to the concept lattice of the original context.

In this section, we focus on a particular clarification of $R$-contexts which applies only to attributes of $R$-contexts. In addition, the procedure of clarification we introduce here produces an $R$-context as a result, i.e., we cope with particular form of attributes which consist of a flag and a set of attributes of the original context, see Definition 3.1. The basic idea is the same as in [7], we produce a new $R$-context by putting together identical columns of the corresponding data table.

For any $R$-context $\mathbb{K}^\sharp = \langle X^\sharp, Y^\sharp, I^\sharp \rangle$ which is derived from $\mathbb{K}$, we can consider a binary relation $\equiv_{\mathbb{K}^\sharp}$ on $Y^\sharp$ such that $y_1 \equiv_{\mathbb{K}^\sharp} y_2$ iff $\{y_1\}^{\downarrow_{\mathbb{K}^\sharp}} = \{y_2\}^{\downarrow_{\mathbb{K}^\sharp}}$. Hence, $y_1 \equiv_{\mathbb{K}^\sharp} y_2$ if columns of the data table corresponding to $\mathbb{K}^\sharp$ given by attributes $y_1$ and $y_2$ are the same. Obviously, $\equiv_{\mathbb{K}^\sharp}$ is an equivalence relation and thus we may consider the corresponding quotient set $Y^\sharp / \equiv_{\mathbb{K}^\sharp}$, denoting the equivalence class of $\equiv_{\mathbb{K}^\sharp}$ containing $y \in Y^\sharp$ by $[y]_{\equiv_{\mathbb{K}^\sharp}}$. Under this notation, we introduce the following notion:

**Definition 3.6.** For any $R$-context $\mathbb{K}^\sharp = \langle X^\sharp, Y^\sharp, I^\sharp \rangle$, we define $X^\complement$, $Y^\complement$, $I^\complement$ as follows:

(i) $X^\complement = X^\sharp$;

(ii) $Y^\complement = \big\{ \big\langle \sum\{n \in \mathbb{N}_0 \mid \langle n, B \rangle \in [y]_{\equiv_{\mathbb{K}^\sharp}}\}, \big\lfloor [y]_{\equiv_{\mathbb{K}^\sharp}} \big\rfloor \big\rangle \mid y \in Y^\sharp \big\}$,

(iii) $I^\complement = \big\{ \langle x, \langle n, B \rangle \rangle \in X^\complement \times Y^\complement \mid$ there is $n' \leq n$ and $B' \subseteq B$ such that $\langle x, \langle n', B' \rangle \rangle \in I^\sharp \big\}$.

Moreover, $\mathbb{K}^\complement = \langle X^\complement, Y^\complement, I^\complement \rangle$ is called a *clarified $R$-context* (*which results from* $\mathbb{K}^\sharp$). ■

**Remark 3.7.** Examining (ii) of Definition 3.6, the set $Y^\complement$ of attributes contains pairs $\langle n, B \rangle$, where $n$ is a numerical flag which results by taking a sum of flags of all attributes in a single equivalence class of $\equiv_{\mathbb{K}^\sharp}$. Analogously, $B$ is a union of sets of original attributes which can be found in attributes from the same equivalence class. While the idea behind taking unions of sets of attributes is clear since attributes indistinguishable under $\equiv_{\mathbb{K}^\sharp}$ are grouped together, the intuitive meaning of taking a sum of flags may not be clear at this point. The informal explanation is the following: in $\langle n, B \rangle \in Y^\sharp$, the number $n$ says that "exactly $n$ of the original attributes from $B$ are not permitted to be used (at certain level of computation)". Thus, if we group attributes together, the numbers of attributes which are not permitted are added since the sets of attributes are disjoint. A formal justification will follow in Section 4.

The following assertion shows basic properties of clarified $R$-contexts.

**Lemma 3.8.** Each clarified $R$-context $\mathbb{K}^\complement$ is a well-defined $R$-context. Moreover, for $\mathbb{K}^\complement$ we have that $\equiv_{\mathbb{K}^\complement}$ is identity. As a consequence, $(\mathbb{K}^\complement)^\complement = \mathbb{K}^\complement$.

**Proof:**
It suffices to check requirements (ii)–(iv) of Definition 3.1. It is immediate that (ii) is satisfied since $Y^\sharp/\equiv_{\mathbb{K}^\sharp}$ consists of pairwise disjoint and nonempty classes which define attributes in $\mathbb{K}^\complement$. Furthermore, (iii) is satisfied because for any $x \in X^\complement = X^\sharp$, $\langle n, B\rangle \in Y^\complement$, and $y_1, y_2 \in B$, there are $\langle n_1, B_1\rangle \in Y^\sharp$ and $\langle n_2, B_2\rangle \in Y^\sharp$ such that $y_1 \in B_1$, $y_2 \in B_2$, and $\langle n_1, B_1\rangle \equiv_{\mathbb{K}^\sharp} \langle n_2, B_2\rangle$. Therefore, $\langle x, y_1\rangle \in I$ iff $\langle x, \langle n_1, B_1\rangle\rangle \in I^\sharp$ iff $\langle x, \langle n_2, B_2\rangle\rangle \in I^\sharp$ iff $\langle x, y_2\rangle \in I$, i.e. (iii) is satisfied by $\mathbb{K}^\complement$. In order to show (iv), observe that by Definition 3.6, $\langle x, \langle n, B\rangle\rangle \in I^\complement$ iff there is $n' \leq n$ and $B' \subseteq B$ such that $\langle n', B'\rangle \in Y^\sharp$ and $\langle x, \langle n', B'\rangle\rangle \in I^\sharp$. Taking into account $\equiv_{\mathbb{K}^\sharp}$, $\langle x, \langle n, B\rangle\rangle \in I^\complement$ iff for any $\langle n', B'\rangle \in Y^\sharp$ such that $n' \leq n$ and $B' \subseteq B$, we have $\langle x, \langle n', B'\rangle\rangle \in I^\sharp$. Using Definition 3.1 (iv) which holds for $\mathbb{K}^\sharp$, the latter is true iff for any $\langle n', B'\rangle \in Y^\sharp$ such that $n' \leq n$ and $B' \subseteq B$, we have $\langle x, y\rangle \in I$ for all $y \in B'$, i.e., $\langle x, y\rangle \in I$ for all $y \in B$ because $B$ is a union of all such $B'$s, proving (iv) of Definition 3.1 for $\mathbb{K}^\complement$. The remaining claims follow easily. □

Table 2. Clarified $R$-context $\mathbb{K}^\complement$ (left) and $\mathbb{K}^\complement$ with sorted attributes (right).

| $\mathbb{K}^\complement$ | $\langle 0,\{1,4\}\rangle$ | $\langle 1,\{2,7\}\rangle$ | $\langle 0,\{3\}\rangle$ | $\langle 0,\{6\}\rangle$ |
|---|---|---|---|---|
| $b$ | | × | × | |
| $d$ | × | × | | × |
| $e$ | × | | | |
| $f$ | | × | × | |

| $\mathbb{K}^\complement$ | $\langle 0,\{6\}\rangle$ | $\langle 0,\{1,4\}\rangle$ | $\langle 0,\{3\}\rangle$ | $\langle 1,\{2,7\}\rangle$ |
|---|---|---|---|---|
| $b$ | | | × | × |
| $d$ | × | × | | × |
| $e$ | | × | | |
| $f$ | | | × | × |

**Example 3.9.** Consider $\mathbb{K}$ and $\mathbb{K}^\sharp$ from Table 1. Then, the clarified $R$-context which results from $\mathbb{K}^\sharp$ is depicted in Table 2. Notice that only original attributes that have been put together are $\langle 1,\{2\}\rangle$ and $\langle 0,\{7\}\rangle$. Since the flags are added, the flag of the resulting attribute $\langle 1,\{2,7\}\rangle$ is equal to 1. Flags of the other attributes remain zero.

**Remark 3.10.** Notice that in our approach, we do not consider clarification of objects, i.e., $\mathbb{K}^\complement$ may contain several objects having the same attributes. Clarification of objects is not used in the subsequent algorithm because in our approach it would not reduce the number of concepts computed multiple times and is therefore omitted.

## 3.3. Attribute Sorting

The algorithm described in Section 4 relies on attribute sorting. In particular, for each $R$-context $\mathbb{K}^\sharp = \langle X^\sharp, Y^\sharp, I^\sharp\rangle$, we consider a partial order $\leq^\sharp$ on $Y^\sharp$ such that for any $y_1, y_2 \in Y^\sharp$, $y_1 \leq^\sharp y_2$ implies $|\{y_1\}^{\downarrow_{\mathbb{K}^\sharp}}| \leq |\{y_2\}^{\downarrow_{\mathbb{K}^\sharp}}|$. In general, $\leq^\sharp$ is not a linear order (not even in the case of clarified $R$-contexts) but it can be extended to a linear order by a well-known procedure of topological sorting.

In next sections, we do not use $\leq^\sharp$ directly. Instead, we assume that we have a bijective map which assigns to each attribute from $Y^\sharp$ its numerical index which represents a position in an ordered list of attributes which are sorted according to (a linear extension of) $\leq^\sharp$. In a more detail, for any $R$-context $\mathbb{K}^\sharp = \langle X^\sharp, Y^\sharp, I^\sharp\rangle$ we consider a bijective map $f\colon Y^\sharp \to \{0, \ldots, |Y^\sharp|-1\}$ such that, for any $y_1, y_2 \in Y^\sharp$,

$$\text{if } f(y_1) \leq f(y_2), \text{ then } |\{y_1\}^{\downarrow_{\mathbb{K}^\sharp}}| \leq |\{y_2\}^{\downarrow_{\mathbb{K}^\sharp}}|. \tag{4}$$

The inverse $f^{-1}$ of $f$ is a map which assigns to each index $j \in \{0, \ldots, |Y^\sharp| - 1\}$ the corresponding attribute $f^{-1}(j) \in Y^\sharp$.

**Example 3.11.** Any $R$-context can be depicted with attributes sorted according to $f$. That is, if $f(y_1) < f(y_2)$, then $y_1$ is depicted before $y_2$. Table 2 (right) shows the results if we apply this idea to the $R$-context $\mathbb{K}^\complement$ from Table 2 (left). Note that in this particular case, there are two ways to define $f$ since attributes $\langle 0, \{1, 4\}\rangle$ and $\langle 0, \{3\}\rangle$ have the same support. In such situations, we always consider an arbitrary (but fixed) $f$ for the same $R$-context.

**Remark 3.12.** In [13], we have investigated the influence of attribute sorting for the CbO family of algorithms. From this point of view, we have considered the same ordering of attributes according to their support. An important distinguishing feature of the present approach is that we do not consider single $\leq^\sharp$ (i.e., a single $f$) during the computation. Instead, during the computation, we successively reduce the initial $R$-context and after each reduction, we determine new $f$ which applies to the reduced $R$-context.

## 3.4. Context Reduction

We now describe a particular reduction operation on $R$-contexts which utilizes operations on $R$-contexts defined in previous sections. The algorithm described in Section 4 uses this operation directly to reduce the problem of computing formal concepts of an $R$-context to the problem of computing formal concepts of several smaller $R$-contexts. From this point of view, the proposed algorithm follows the usual *divide et impera* scheme of decomposing an instance of a problem into several instances of the same problem of smaller sizes which in turn leads to a concise implementation of the algorithm by a recursive procedure.

The input for reduction is a clarified $R$-context $\mathbb{K}^\sharp = \langle X^\sharp, Y^\sharp, I^\sharp\rangle$ and a formal concept $\langle C, D\rangle$ in $\mathbb{K}^\sharp$ whose intent is nonempty, i.e. $D \neq \emptyset$. For $\mathbb{K}^\sharp$, we assume that we are given a bijective map satisfying (4) which determines the order of attributes in $\mathbb{K}^\sharp$. Since $D$ is nonempty, we can denote by $\min(D)$ the least attribute from $D$ with respect to the order given by $f$, i.e., $\min(D) \in D$ such that $f(\min(D)) \leq f(y)$ for all $y \in D$. Using this notation, we define the following notion:

**Definition 3.13.** For any $R$-context $\mathbb{K}^\sharp = \langle X^\sharp, Y^\sharp, I^\sharp\rangle$, $C \subseteq X^\sharp$, $\emptyset \neq D \subseteq Y^\sharp$ such that $C^{\uparrow_{\mathbb{K}^\sharp}} = D$ and $D^{\downarrow_{\mathbb{K}^\sharp}} = C$, we define $X^\mathfrak{R}$, $Y^\mathfrak{R}$, and $I^\mathfrak{R}$ as follows:

(i) $X^\mathfrak{R} = C$;

(ii) $Y^\mathfrak{R} = \{\mathrm{Attr}(y) \mid y \in Y^\sharp \text{ and } y \notin D\}$, where $\mathrm{Attr}(y) \in \mathbb{N}_0 \times 2^Y$ is defined by

$$\mathrm{Attr}(\langle n, B\rangle) = \begin{cases} \langle |B|, B\rangle, & \text{if } n = 0 \text{ and } f(\langle n, B\rangle) < f(\min(D)), \\ \langle n, B\rangle, & \text{otherwise,} \end{cases} \tag{5}$$

for any $\langle n, B\rangle \in Y^\sharp$;

(iii) $I^\mathfrak{R} = \big\{\langle x, \langle n, B\rangle\rangle \in X^\mathfrak{R} \times Y^\mathfrak{R} \mid \text{ there is } n' \leq n \text{ such that } \langle x, \langle n', B\rangle\rangle \in I^\sharp\big\}$. ■

**Remark 3.14.** One can easily see that $\mathbb{K}^\mathfrak{R} = \langle X^\mathfrak{R}, Y^\mathfrak{R}, I^\mathfrak{R}\rangle$ as defined in Definition 3.13 is an $R$-context with objects taken from $C$, attributes being derived from attributes in $Y^\sharp$ which are not present in $D$. Note

that for each $\langle n, B\rangle \in Y^\sharp$ which is not in $D$, $Y^\mathfrak{R}$ contains an attribute $\langle n', B\rangle \in Y^\mathfrak{R}$, where $n'$ is a new flag. The value of the flag is either the same if $f(\langle n, B\rangle)$ is greater or equal to $f(\min(D))$ or the flag is equal to the size of $B$. In other words, if $\langle n, B\rangle$ is behind $\min(D)$ in terms of the order of attributes, the flag is not updated. The most important part of the flag update is that an attribute $\langle 0, B\rangle \in Y^\sharp$ will be given a nonzero flag in $Y^\mathfrak{R}$ if it is not in $D$ and if it stays before $\min(D)$ in terms of the order of attributes.

In general, $\mathbb{K}^\mathfrak{R} = \langle X^\mathfrak{R}, Y^\mathfrak{R}, I^\mathfrak{R}\rangle$ can contain two or more indistinguishable attributes (equal columns in the corresponding data table), i.e., $\mathbb{K}^\mathfrak{R}$ may not be clarified in sense of Definition 3.6. The algorithm described in the next section relies on reduction and clarification of $R$-contexts, we therefore introduce the following notation:

**Definition 3.15.** If $\mathbb{K}^\mathfrak{R}$ results from $\mathbb{K}^\sharp$ using $C$ and $D$ in sense of Definition 3.13 and if $\mathbb{K}^\mathfrak{C}$ is a clarification of $\mathbb{K}^\mathfrak{R}$ in sense of Definition 3.6, then $\mathbb{K}^\mathfrak{C}$ will be denoted by REDUCE$(\mathbb{K}^\sharp, C, D)$. ■

Table 3. Context from Definition 3.13 (left), result of REDUCE (middle), and its concise representation (right).

| $\mathbb{K}^\mathfrak{R}$ | $\langle 1,\{6\}\rangle$ | $\langle 0,\{3\}\rangle$ | $\langle 1,\{2,7\}\rangle$ |
|---|---|---|---|
| $d$ | × | | × |
| $e$ | | | |

| $\mathbb{K}^\mathfrak{R}$ | $\langle 0,\{3\}\rangle$ | $\langle 2,\{2,6,7\}\rangle$ |
|---|---|---|
| $d$ | | × |
| $e$ | | |

| $\mathbb{K}^\mathfrak{R}$ | $\{3\}$ | $\{2,6,7\}$ |
|---|---|---|
| $d$ | | × |
| $e$ | | |

**Example 3.16.** Consider $R$-context from Table 2 (right). For $C = \{d, e\}$, and $D = \{\langle 0, \{1, 4\}\rangle\}$, the $R$-context $\mathbb{K}^\mathfrak{R}$ specified in Definition 3.13 is depicted in Table 3 (left). Notice that during the reduction, attribute $\langle 1, \{6\}\rangle$ was given a nonzero flag since its position according to $f$ was before that of attribute $\langle 0, \{1, 4\}\rangle$ in the original $R$-context. Table 3 (middle) represents a clarified version of $\mathbb{K}^\mathfrak{R}$ with attributes sorted according to their supports. Hence, the middle table represents the result of REDUCE. Table 3 (right) is a concise representation of the same $R$-context in which columns corresponding to attributes with nonzero flags are highlighted as gray (the descriptions of attributes then contain just sets of original attributes, the numerical flags are omitted).

## 4. Algorithm

In this section, we describe the proposed algorithm for computing formal concepts. The main part of the algorithm is a recursive procedure COMPUTE from Algorithm 1. The procedure accepts as its argument a clarified $R$-context and during the computation it calls an auxiliary procedure CLOSURE from Algorithm 2.

When invoked with $\mathbb{K}^\sharp$, procedure COMPUTE proceeds as follows. First, it stores a tuple which consists of the set of objects $X^\sharp$ and INT$(\mathbb{K}^\sharp, Y)$, where

$$\text{INT}(\mathbb{K}^\sharp, Y) = Y \setminus \lfloor Y^\sharp \rfloor. \tag{6}$$

Recall that $\lfloor \cdots \rfloor$ is defined by (1). Thus, INT$(\mathbb{K}^\sharp, Y) = Y \setminus \bigcup\{B \subseteq Y \mid \langle n, B\rangle \in Y^\sharp\}$. Then, the procedure goes over all attributes in $Y^\sharp$ with zero flags (see lines 2 and 3 of Algorithm 1). For each such

**Algorithm 1:** Procedure COMPUTE$(\mathbb{K}^\sharp)$

```
1  store ⟨X♯, INT(K♯, Y)⟩;
2  for ⟨n, B⟩ ∈ Y♯ do
3      if n = 0 then
4          set ⟨C, D⟩ to CLOSURE(K♯, ⟨n, B⟩);
5          if Σ{n ∈ ℕ0 | ⟨n, B⟩ ∈ D} = 0 then
6              COMPUTE(REDUCE(K♯, C, D));
7          end
8      end
9  end
10 return
```

**Algorithm 2:** Procedure CLOSURE$(\mathbb{K}^\sharp, \langle n, B\rangle)$

```
1  C = {x ∈ X♯ | ⟨x, ⟨n, B⟩⟩ ∈ I♯};
2  D = {y ∈ Y♯ | f(⟨n, B⟩) ≤ f(y)};
3  for x ∈ C do
4      for y ∈ D do
5          if ⟨x, y⟩ ∉ I♯ then
6              remove y from D;
7          end
8      end
9  end
10 return ⟨C, D⟩
```

attribute $\langle n, B\rangle$, the procedure invokes CLOSURE and the result of invocation is stored in $\langle C, D\rangle$. An easy inspection of the pseudocode in Algorithm 2 shows that the result of calling CLOSURE$(\mathbb{K}^\sharp, \langle n, B\rangle)$ is a formal concept in $\mathbb{K}^\sharp$ generated by attribute $\langle n, B\rangle$, i.e., $C = \{\langle n, B\rangle\}^{\downarrow_{\mathbb{K}^\sharp}}$ and $D = C^{\uparrow_{\mathbb{K}^\sharp}}$. Notice that Algorithm 2 utilizes attribute sorting together with the fact that $\mathbb{K}^\sharp$ is clarified. In that case, all attributes which belong to $D$ must have their indices strictly greater than or equal to $f(\langle n, B\rangle)$. This observation has already been made in [13].

Next step of Algorithm 1 is a canonicity test which succeeds iff all flags in $D$ (computed in the previous step) are zero, see line 5. In the case of success, COMPUTE invokes itself with reduced (and clarified) formal context which results from $\mathbb{K}^\sharp$, see line 6. Otherwise, the algorithm continues with another attribute. When all attributes are processed, the invocation of COMPUTE for $\mathbb{K}^\sharp$ is left.

For input formal context $\mathbb{K} = \langle X, Y, I\rangle$, the first invocation of COMPUTE can be described by the following consecutive steps:

1. take an initial $R$-context $\mathbb{K}^\sharp = \langle X^\sharp, Y^\sharp, I^\sharp\rangle$ derived from $\mathbb{K}$;
2. determine a clarified $R$-context $\mathbb{K}^\complement = \langle X^\complement, Y^\complement, I^\complement\rangle$ which results from $\mathbb{K}^\sharp$;
3. if $|\{\langle n, B\rangle\}^{\downarrow_{\mathbb{K}^\complement}}| < |X^\complement|$ for all $\langle n, B\rangle \in Y^\complement$ then call COMPUTE$(\mathbb{K}^\complement)$.

4. if there is $\langle n, B\rangle \in Y^{\complement}$ such that $|\{\langle n, B\rangle\}^{\downarrow_{\mathbb{K}^{\complement}}}| = |X^{\complement}|$, then call COMPUTE$(\mathbb{K}^*)$, where $\mathbb{K}^* = \langle X^*, Y^*, I^*\rangle$ with $X^* = X^{\complement}$, $Y^* = Y^{\complement} \setminus \{\langle n, B\rangle\}$, and $I^* = I^{\complement} \cap (X^* \times Y^*)$.

In other words, $\mathbb{K}$ is transformed into an $R$-context and clarified. If the resulting $R$-context contains an attribute shared by all objects (notice that since it is clarified, such an attribute is at most one), it is removed from the $R$-context. Then, COMPUTE is invoked with such $R$-context as an input. In Section 6, we shall prove that the algorithm is sound, i.e., with input data of this form, it stores all formal concepts, each of them exactly once.

**Remark 4.1.** Notice that the canonicity test is expressed using a sum, see line 5 of Algorithm 1. One can easily see that we might as well use "logical or" provided that all flags are assigned values 0 and 1, only. This can be achieved by slight modifications of $\mathrm{Attr}(\langle n, B\rangle)$ which appears in Definition 3.13 and $Y^{\complement}$ defined in Definition 3.6. Indeed, the numerical value of the flag is not as important for the algorithm. The important fact is whether at least one of the attributes in intent $D$ has nonzero flag, see Algorithm 1.

Table 4. Illustrative formal context

| $\mathbb{K}$ | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| $a$ | | × | × | × | | |
| $b$ | × | × | | | | × |
| $c$ | | × | | × | × | |
| $d$ | × | | × | | × | |

## 5. Illustrative Example

Before we investigate properties of Algorithm 1, we show here an illustrative running example in which we demonstrate how COMPUTE behaves for particular input data. This illustration is useful for getting first (informal) insight into the algorithm. Consider an input formal context $\mathbb{K} = \langle X, Y, I\rangle$ with objects $X = \{a, b, c, d\}$, attributes $Y = \{0, 1, 2, 3, 4, 5\}$, and $I \subseteq X \times Y$ as in Table 4. One can check that $\mathbb{K}$ has 11 formal concepts, namely:

$$
\begin{aligned}
R_1 &= \langle\{a, b, c, d\}, \emptyset\rangle, & R_5 &= \langle\{d\}, \{0, 2, 4\}\rangle, & R_9 &= \langle\{c\}, \{1, 3, 4\}\rangle,\\
R_2 &= \langle\{b\}, \{0, 1, 5\}\rangle, & R_6 &= \langle\{a, d\}, \{2\}\rangle, & R_{10} &= \langle\{c, d\}, \{4\}\rangle,\\
R_3 &= \langle\emptyset, \{0, 1, 2, 3, 4, 5\}\rangle, & R_7 &= \langle\{a\}, \{1, 2, 3\}\rangle, & R_{11} &= \langle\{a, b, c\}, \{1\}\rangle.\\
R_4 &= \langle\{b, d\}, \{0\}\rangle, & R_8 &= \langle\{a, c\}, \{1, 3\}\rangle,
\end{aligned}
$$

Algorithm 1 proceeds for $\mathbb{K}$ as follows. First, an initial and clarified $R$-context is created, denote it by $\mathbb{K}_1^\sharp$. Since in $\mathbb{K}$ all attributes are distinct and there is no attibute which is shared by all objects, $\mathbb{K}_1^\sharp$ is directly passed to COMPUTE as the initial argument. The initial $R$-context is depicted in Figure 1 (top).

The execution of COMPUTE proceeds with selecting an attribute from $Y_1^\sharp$, computing the closure and reduction $\mathbb{K}_2^\sharp$, and recursive invocation of COMPUTE:

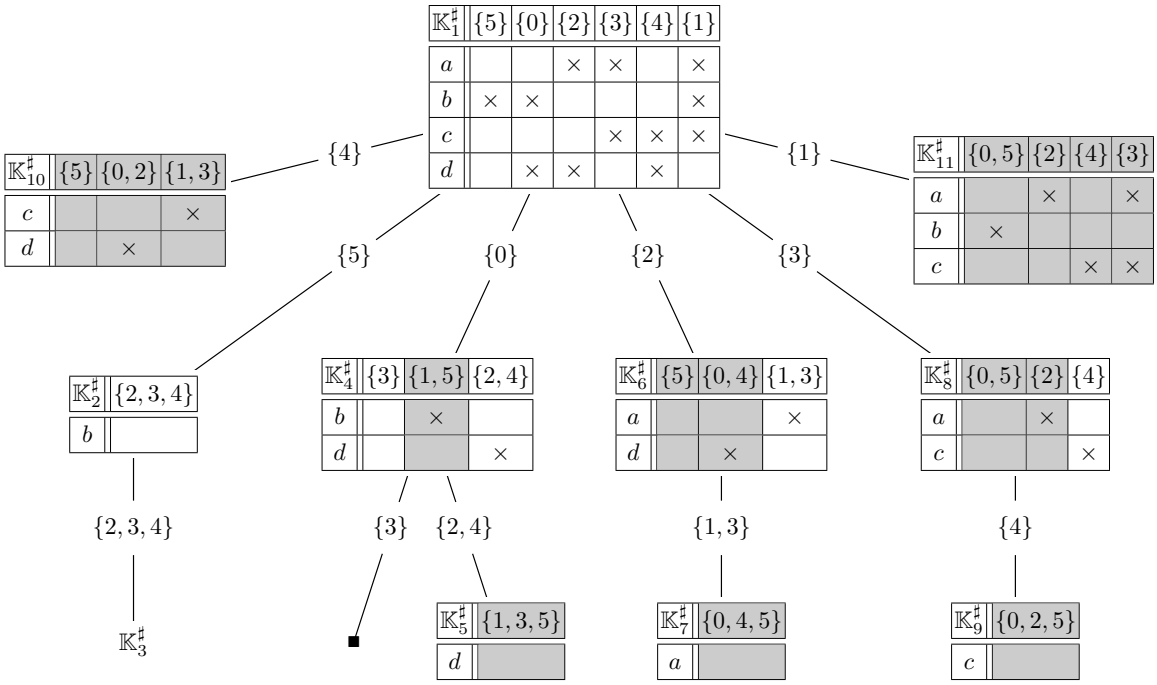


Figure 1. $R$-contexts produced by Algorithm 1 during computation.

line 1: **store** $\langle X_1^\sharp, \textsc{Int}(\mathbb{K}_1^\sharp, Y)\rangle = \langle \boldsymbol{\{a,b,c,d\}, \emptyset}\rangle = \boldsymbol{R_1}$
line 4: set $\langle C_2, D_2\rangle$ to $\textsc{Closure}(\mathbb{K}_1^\sharp, \langle 0, \{5\}\rangle) = \langle \{b\}, \{\langle 0, \{5\}\rangle, \langle 0, \{0\}\rangle, \langle 0, \{1\}\rangle\}\rangle$
line 5: success for $\langle C_2, D_2\rangle = \langle \{b\}, \{\langle 0, \{5\}\rangle, \langle 0, \{0\}\rangle, \langle 0, \{1\}\rangle\}\rangle$ because all flags are $0$
line 6: call $\textsc{Compute}(\mathbb{K}_2^\sharp)$ for $\mathbb{K}_2^\sharp = \textsc{Reduce}(\mathbb{K}_1^\sharp, C_2, D_2)$

Notice that in Figure 1, the recursive invocation is depicted by an $R$-context $\mathbb{K}_2^\sharp$ connected with $\mathbb{K}_1^\sharp$ with an edge labeled by $\{5\}$ which is the original set of attributes present in attribute $\langle 0, \{5\}\rangle \in Y_1^\sharp$. Moreover, the computation continues as follows:

line 1: **store** $\langle X_2^\sharp, \textsc{Int}(\mathbb{K}_2^\sharp, Y)\rangle = \langle \boldsymbol{\{b\}, \{0,1,5\}}\rangle = \boldsymbol{R_2}$
line 4: set $\langle C_3, D_3\rangle$ to $\textsc{Closure}(\mathbb{K}_2^\sharp, \langle 0, \{2,3,4\}\rangle) = \langle \emptyset, \{\langle 0, \{2,3,4\}\rangle\}\rangle$
line 5: success for $\langle C_3, D_3\rangle = \langle \emptyset, \{\langle 0, \{2,3,4\}\rangle\}\rangle$ because all flags are $0$
line 6: call $\textsc{Compute}(\mathbb{K}_3^\sharp)$ for $\mathbb{K}_3^\sharp = \textsc{Reduce}(\mathbb{K}_2^\sharp, C_3, D_3)$
  line 1: **store** $\langle X_3^\sharp, \textsc{Int}(\mathbb{K}_3^\sharp, Y)\rangle = \langle \boldsymbol{\emptyset, \{0,1,2,3,4,5\}}\rangle = \boldsymbol{R_3}$
  $\perp$ return from invocation of $\textsc{Compute}$ for $\mathbb{K}_3^\sharp$
$\perp$ return from invocation of $\textsc{Compute}$ for $\mathbb{K}_2^\sharp$

Notice that since $\mathbb{K}_3^\sharp$ is a trivial context with empty sets of objects and attributes, the invocation of $\textsc{Compute}$ has immediatelly returned after storing $R_3$ because the iteration of the for-loop is trivially done for empty $Y_3^\sharp$. Next, the computation resumes in the first invocation of $\textsc{Compute}$ considering next attribute:

line 4: set $\langle C_4, D_4\rangle$ to CLOSURE($\mathbb{K}_1^\sharp, \langle 0, \{0\}\rangle$) = $\langle\{b, d\}, \{\langle 0, \{0\}\rangle\}\rangle$
line 5: success for $\langle C_4, D_4\rangle = \langle\{b, d\}, \{\langle 0, \{0\}\rangle\}\rangle$ because all flags are 0
line 6: call COMPUTE($\mathbb{K}_4^\sharp$) for $\mathbb{K}_4^\sharp$ = REDUCE($\mathbb{K}_1^\sharp, C_4, D_4$)
  line 1: **store** $\langle X_4^\sharp, \text{INT}(\mathbb{K}_4^\sharp, Y)\rangle = \boldsymbol{\langle\{b, d\}, \{0\}\rangle = R_4}$
  line 4: set $\langle C_5, D_5\rangle$ to CLOSURE($\mathbb{K}_4^\sharp, \langle 0, \{3\}\rangle$) = $\langle\emptyset, \{\langle 0, \{3\}\rangle, \langle 1, \{1, 5\}\rangle, \langle 0, \{2, 4\}\rangle\}\rangle$
  line 5: failure for $\langle C_5, D_5\rangle = \langle\emptyset, \{\langle 0, \{3\}\rangle, \langle 1, \{1, 5\}\rangle, \langle 0, \{2, 4\}\rangle\}\rangle$ because $\langle 1, \{1, 5\}\rangle \in D_5$

At this point, the canonicity test has failed. Therefore, the algorithm does not continue with $\langle C_5, D_5\rangle$ which in fact determines formal concept $R_3$ that has been computed and processed before. This is the only point where the canonicity test fails in this example and where a concept is computed more than once. Notice that it is not the case that $R_3$ is computed as such, the algorithm has computed $\langle C_5, D_5\rangle$ but anyhow, $\langle C_5, D_5\rangle$ would normally be used to determine $R_3$, i.e. we can consider $R_3$ to be computed twice. In Figure 1 the situation is depicted by a black square node labeled by $R_3$. After this point, the computation continues as follows (without further comments):

  line 4: set $\langle C_5, D_5\rangle$ to CLOSURE($\mathbb{K}_4^\sharp, \langle 0, \{2, 4\}\rangle$) = $\langle\{d\}, \{\langle 0, \{2, 4\}\rangle\}\rangle$
  line 5: success for $\langle C_5, D_5\rangle = \langle\{d\}, \{\langle 0, \{2, 4\}\rangle\}\rangle$ because all flags are 0
  line 6: call COMPUTE($\mathbb{K}_5^\sharp$) for $\mathbb{K}_5^\sharp$ = REDUCE($\mathbb{K}_4^\sharp, C_5, D_5$)
    line 1: **store** $\langle X_5^\sharp, \text{INT}(\mathbb{K}_5^\sharp, Y)\rangle = \boldsymbol{\langle\{d\}, \{0, 2, 4\}\rangle = R_5}$
    $\perp$ return from invocation of COMPUTE for $\mathbb{K}_5^\sharp$
  $\perp$ return from invocation of COMPUTE for $\mathbb{K}_4^\sharp$
line 4: set $\langle C_6, D_6\rangle$ to CLOSURE($\mathbb{K}_1^\sharp, \langle 0, \{2\}\rangle$) = $\langle\{a, d\}, \{\langle 0, \{2\}\rangle\}\rangle$
line 5: success for $\langle C_6, D_6\rangle = \langle\{a, d\}, \{\langle 0, \{2\}\rangle\}\rangle$ because all flags are 0
line 6: call COMPUTE($\mathbb{K}_6^\sharp$) for $\mathbb{K}_6^\sharp$ = REDUCE($\mathbb{K}_1^\sharp, C_6, D_6$)
  line 1: **store** $\langle X_6^\sharp, \text{INT}(\mathbb{K}_6^\sharp, Y)\rangle = \boldsymbol{\langle\{a, d\}, \{2\}\rangle = R_6}$
  line 4: set $\langle C_7, D_7\rangle$ to CLOSURE($\mathbb{K}_6^\sharp, \langle 0, \{1, 3\}\rangle$) = $\langle\{a\}, \{\langle 0, \{1, 3\}\rangle\}\rangle$
  line 5: success for $\langle C_7, D_7\rangle = \langle\{a\}, \{\langle 0, \{1, 3\}\rangle\}\rangle$ because all flags are 0
  line 6: call COMPUTE($\mathbb{K}_7^\sharp$) for $\mathbb{K}_7^\sharp$ = REDUCE($\mathbb{K}_6^\sharp, C_7, D_7$)
    line 1: **store** $\langle X_7^\sharp, \text{INT}(\mathbb{K}_7^\sharp, Y)\rangle = \boldsymbol{\langle\{a\}, \{1, 2, 3\}\rangle = R_7}$
    $\perp$ return from invocation of COMPUTE for $\mathbb{K}_7^\sharp$
  $\perp$ return from invocation of COMPUTE for $\mathbb{K}_6^\sharp$
line 4: set $\langle C_8, D_8\rangle$ to CLOSURE($\mathbb{K}_1^\sharp, \langle 0, \{3\}\rangle$) = $\langle\{a, c\}, \{\langle 0, \{3\}\rangle, \langle 0, \{1\}\rangle\}\rangle$
line 5: success for $\langle C_8, D_8\rangle = \langle\{a, c\}, \{\langle 0, \{3\}\rangle, \langle 0, \{1\}\rangle\}\rangle$ because all flags are 0
line 6: call COMPUTE($\mathbb{K}_8^\sharp$) for $\mathbb{K}_8^\sharp$ = REDUCE($\mathbb{K}_1^\sharp, C_8, D_8$)
  line 1: **store** $\langle X_8^\sharp, \text{INT}(\mathbb{K}_8^\sharp, Y)\rangle = \boldsymbol{\langle\{a, c\}, \{1, 3\}\rangle = R_8}$
  line 4: set $\langle C_9, D_9\rangle$ to CLOSURE($\mathbb{K}_8^\sharp, \langle 0, \{4\}\rangle$) = $\langle\{c\}, \{\langle 0, \{4\}\rangle\}\rangle$
  line 5: success for $\langle C_9, D_9\rangle = \langle\{c\}, \{\langle 0, \{4\}\rangle\}\rangle$ because all flags are 0
  line 6: call COMPUTE($\mathbb{K}_9^\sharp$) for $\mathbb{K}_9^\sharp$ = REDUCE($\mathbb{K}_8^\sharp, C_9, D_9$)
    line 1: **store** $\langle X_9^\sharp, \text{INT}(\mathbb{K}_9^\sharp, Y)\rangle = \boldsymbol{\langle\{c\}, \{1, 3, 4\}\rangle = R_9}$
    $\perp$ return from invocation of COMPUTE for $\mathbb{K}_9^\sharp$
  $\perp$ return from invocation of COMPUTE for $\mathbb{K}_8^\sharp$
line 4: set $\langle C_{10}, D_{10}\rangle$ to CLOSURE($\mathbb{K}_1^\sharp, \langle 0, \{4\}\rangle$) = $\langle\{c, d\}, \{\langle 0, \{4\}\rangle\}\rangle$

line 5: success for $\langle C_{10}, D_{10}\rangle = \langle\{c, d\}, \{\langle 0, \{4\}\rangle\}\rangle$ because all flags are $0$
line 6: call COMPUTE($\mathbb{K}^\sharp_{10}$) for $\mathbb{K}^\sharp_{10} = $ REDUCE($\mathbb{K}^\sharp_1, C_{10}, D_{10}$)
  line 1: **store** $\langle X^\sharp_{10}, \text{INT}(\mathbb{K}^\sharp_{10}, Y)\rangle = \boldsymbol{\langle\{c, d\}, \{4\}\rangle = R_{10}}$
  $\perp$ return from invocation of COMPUTE for $\mathbb{K}^\sharp_{10}$
line 4: set $\langle C_{11}, D_{11}\rangle$ to CLOSURE($\mathbb{K}^\sharp_1, \langle 0, \{1\}\rangle$) = $\langle\{a, b, c\}, \{\langle 0, \{1\}\rangle\}\rangle$
line 5: success for $\langle C_{11}, D_{11}\rangle = \langle\{a, b, c\}, \{\langle 0, \{1\}\rangle\}\rangle$ because all flags are $0$
line 6: call COMPUTE($\mathbb{K}^\sharp_{11}$) for $\mathbb{K}^\sharp_{11} = $ REDUCE($\mathbb{K}^\sharp_1, C_{11}, D_{11}$)
  line 1: **store** $\langle X^\sharp_{11}, \text{INT}(\mathbb{K}^\sharp_{11}, Y)\rangle = \boldsymbol{\langle\{a, b, c\}, \{1\}\rangle = R_{11}}$
  $\perp$ return from invocation of COMPUTE for $\mathbb{K}^\sharp_{11}$
$\perp$ return from invocation of COMPUTE for $\mathbb{K}^\sharp_1$

**Remark 5.1.** It is interesting to compare the presented algorithm with CbO [14, 15, 16] and FCbO [13, 23] in terms of formal concepts which are computed multiple times. In a similar way as in the case of our algorithm, CbO an FCbO are recursively invoked and the computation can therefore be expressed by a corresponding call tree. Figure 2 shows a call tree for both CbO and FCbO applied to input formal context from the example. The bold lines correspond to both CbO and FCbO, the dotted lines correspond only to CbO. The black square nodes labeled by formal concepts represent branches of computation where the concepts are computed but fail the canonicity test. We can see that FCbO computes 7 formal concepts which fail the canonicity test. Thus, several concepts are computed multiple times. Namely, $R_2$ is computed twice, $R_3$ is computed three times, so is $R_5$, and $R_8$ and $R_9$ and both computed twice. Recall that our algorithm computes just a single formal concept twice, so this is an interesting improvement. In the case of CbO, the improvement is even more visible since here the number of computed concepts which fail the canonicity test is $19$. Section 7 shows experimental evaluation of average behavior of our algorithm compared to CbO and FCbO using various data sets which shows an interesting tendency that the numbers of formal concepts computed multiple times by the presented algorithm are much smaller.

## 6. Algorithm Properties and Soundness

In this section, we pay attention to properties of the algorithm and prove its soundness which means that for an input formal context, the algorithm stores each formal concept exactly once. In other words, if a formal concept is calculated several times, the algorithm ensures that it is stored (e.g., printed as an output or stored in an output data structure) at most once; moreover, the algorithm ensures that each formal concept is stored at least once. The two conditions together yield that each formal concept is stored exactly once.

We take the same assumptions as in Section 4. Hence, we assume that $\mathbb{K} = \langle X, Y, I\rangle$ is the input formal context and that COMPUTE is invoked according to the steps described in Section 4.

In order to prove soundness of the algorithm, we first show that each $R$-context which is passed to COMPUTE as an argument during the computation represents a formal context. That is, if one considers line 1 of Algorithm 1, for such an $R$-context, the algorithm stores a couple which is a formal concept in $\mathbb{K}$. Notice that one can easily find an $R$-context for which this is not so. Therefore, we introduce the following notion.

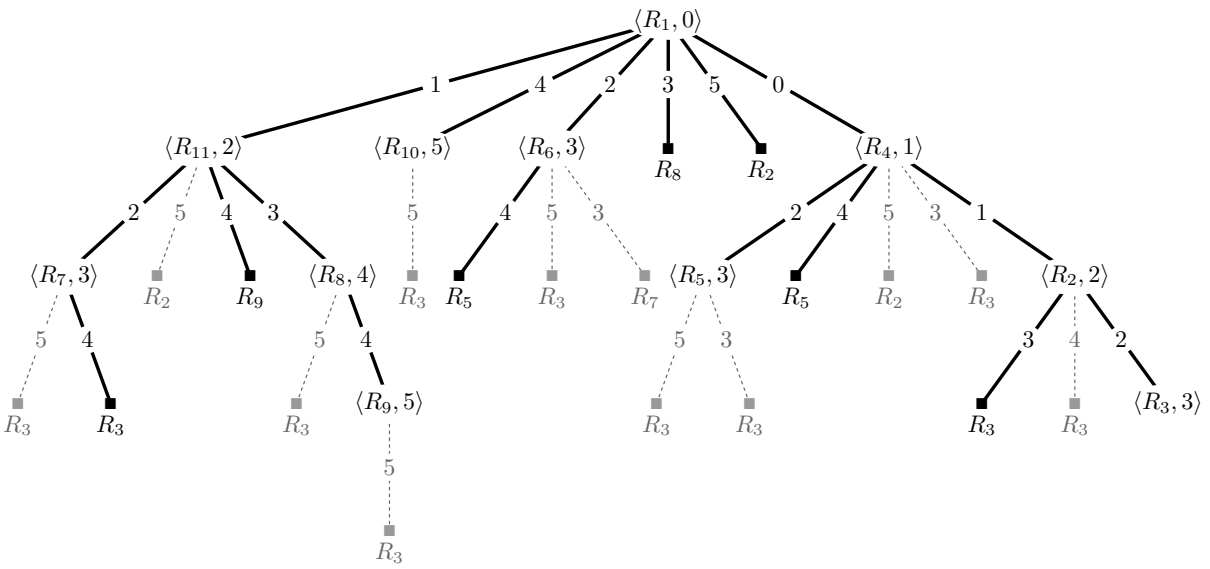


Figure 2. Example of a call tree of FCbO with reduced number of leaf nodes.

**Definition 6.1.** Let $\mathbb{K}^\sharp$ be an $R$-context derived from $\mathbb{K}$. We shall say that $\mathbb{K}^\sharp$ is $\mathbb{K}$-representative if $\langle X^\sharp, \textsc{Int}(\mathbb{K}^\sharp, Y)\rangle$ is a formal concept in $\mathbb{K}$. ■

The definition captures exactly the property that is needed to store (only) formal concepts. The next assertion shows that the property is preserved during consecutive invocations of COMPUTE.

**Lemma 6.2.** Let $\mathbb{K}^\sharp$ be a $\mathbb{K}$-representative $R$-context derived from $\mathbb{K}$ and let $\langle C, D\rangle$ be a formal concept in $\mathbb{K}^\sharp$ with $D \neq \emptyset$. Then, $\textsc{Reduce}(\mathbb{K}^\sharp, C, D)$ is $\mathbb{K}$-representative.

**Proof:**
Denote $\textsc{Reduce}(\mathbb{K}^\sharp, C, D)$ by $\mathbb{K}^\mathfrak{R}$. Since $\mathbb{K}^\sharp$ is $\mathbb{K}$-representative, we have that $(X^\sharp)^{\uparrow_\mathbb{K}} = \textsc{Int}(\mathbb{K}^\sharp, Y)$ and $\textsc{Int}(\mathbb{K}^\sharp, Y)^{\downarrow_\mathbb{K}} = X^\sharp$. Moreover, since $\langle C, D\rangle$ is assumed to be a formal concept in $\mathbb{K}^\sharp$, we have $C^{\uparrow_{\mathbb{K}^\sharp}} = D$ and $D^{\downarrow_{\mathbb{K}^\sharp}} = C$. We now show that $\langle C, \textsc{Int}(\mathbb{K}^\sharp, Y) \cup \lfloor D \rfloor\rangle$ is a formal concept in $\mathbb{K}$. Notice that according to Definition 3.13, this would prove that $\mathbb{K}^\mathfrak{R}$ is $\mathbb{K}$-representative because by Definition 3.13, we have $\textsc{Int}(\mathbb{K}^\mathfrak{R}, Y) = \textsc{Int}(\mathbb{K}^\sharp, Y) \cup \lfloor D \rfloor$.

Using (2), we get $\lfloor D \rfloor = \lfloor C^{\uparrow_{\mathbb{K}^\sharp}} \rfloor \subseteq C^{\uparrow_\mathbb{K}}$. Since $C \subseteq X^\sharp$, we get $\textsc{Int}(\mathbb{K}^\sharp, Y) = (X^\sharp)^{\uparrow_\mathbb{K}} \subseteq C^{\uparrow_\mathbb{K}}$. Putting the inclusions together, we get $\textsc{Int}(\mathbb{K}^\sharp, Y) \cup \lfloor D \rfloor \subseteq C^{\uparrow_\mathbb{K}}$. In order to prove the converse inclusion, it suffices to check that if $y \in C^{\uparrow_\mathbb{K}}$ and $y \notin \textsc{Int}(\mathbb{K}^\sharp, Y)$, then $y \in \lfloor D \rfloor$. If $y \notin \textsc{Int}(\mathbb{K}^\sharp, Y)$, there is $\langle n, B\rangle \in Y^\sharp$ such that $y \in B$. If in addition $y \in C^{\uparrow_\mathbb{K}}$, then $\langle x, y\rangle \in I$ for all $x \in C$. Using Definition 3.1 (iii), $\langle x, y\rangle \in I$ for all $x \in C$ and all $y \in B$, meaning that $\langle x, \langle n, B\rangle\rangle \in I^\sharp$ for all $x \in C$. Hence, $\langle n, B\rangle \in C^{\uparrow_{\mathbb{K}^\sharp}} = D$ which yields $y \in B \subseteq \lfloor D \rfloor$. Altogether, $C^{\uparrow_\mathbb{K}} = \textsc{Int}(\mathbb{K}^\sharp, Y) \cup \lfloor D \rfloor$. Now, using (3), we get $(\textsc{Int}(\mathbb{K}^\sharp, Y) \cup \lfloor D \rfloor)^{\downarrow_\mathbb{K}} = \textsc{Int}(\mathbb{K}^\sharp, Y)^{\downarrow_\mathbb{K}} \cap \lfloor D \rfloor^{\downarrow_\mathbb{K}} = X^\sharp \cap \lfloor D \rfloor^{\downarrow_\mathbb{K}} = D^{\downarrow_{\mathbb{K}^\sharp}} = C$. □

**Corollary 6.3.** All tuples stored during invocations of COMPUTE are formal concepts in $\mathbb{K}$.

**Proof:**
The proof is obvious. Indeed, by induction and using Lemma 6.2, one can check that each $R$-context that is passed to COMPUTE is $\mathbb{K}$-representative. □

Notice that now it becomes apparent why we have removed an attribute shared by all objects from the clarified initial $R$-context (see step 4 described in Section 4). Otherwise, the argument for the first invocation of COMPUTE would not be $\mathbb{K}$-representative, meaning that COMPUTE would store a pair which is not a formal concept (the attribute shared by all objects would not be present in intent of the first stored pair).

The following assertion shows that Algorithm 1 provides a complete search for formal concepts, i.e., each formal concept is stored at least once.

**Lemma 6.4.** During the invocations of COMPUTE, each formal concept in $\mathbb{K}$ is stored at least once.

**Proof:**
For brevity, we denote by $\mathbb{K}_i^\sharp \prec \mathbb{K}_j^\sharp$ the fact that if COMPUTE is invoked with $\mathbb{K}_i^\sharp$, then during its invocation, it invokes itself with $\mathbb{K}_j^\sharp$. Therefore, $\mathbb{K}_j^\sharp$ is equal to $\text{REDUCE}(\mathbb{K}_i^\sharp, C, D)$ for some $C$ and $D$.

Take formal concept $\langle E, F\rangle$ in $\mathbb{K}$. We prove that there is a sequence $\mathbb{K}_1^\sharp \prec \cdots \prec \mathbb{K}_n^\sharp$ of $\mathbb{K}$-representative $R$-contexts derived from $\mathbb{K}$ such that $X_n^\sharp = E$ and $\mathbb{K}_1^\sharp$ is the argument of the first invocation of COMPUTE. This would prove that formal concept $\langle E, F\rangle$ will be stored by COMPUTE invoked with $\mathbb{K}_n^\sharp$.

We construct the sequence as follows. The first element $\mathbb{K}_1^\sharp$ is determined uniquely. Assume that we have constructed first $i$ elements of the sequence and for $\mathbb{K}_i^\sharp$ we have that if $\langle n, B\rangle \in Y_i^\sharp$ and $B \subseteq F$, then $n = 0$. Observe that this property holds for $\mathbb{K}_1^\sharp$ trivially since the flags of all attributes in $Y_1^\sharp$ are all zero. If $X_i^\sharp = E$, we are done. Otherwise, we show that we can choose a $\mathbb{K}$-representative $R$-context $\mathbb{K}_{i+1}^\sharp$ derived from $\mathbb{K}$ such that $\mathbb{K}_i^\sharp \prec \mathbb{K}_{i+1}^\sharp$ for which we have that if $\langle n, B\rangle \in Y_{i+1}^\sharp$ and $B \subseteq F$, then $n = 0$. Thus, if $X_i^\sharp \supset E$, then $\lfloor Y_i^\sharp \rfloor \cap F \neq \emptyset$ because $\mathbb{K}_i^\sharp$ is $\mathbb{K}$-representative. Thus, we can take $\langle n, B\rangle \in Y_i^\sharp$ such that $B \cap F \neq \emptyset$ and $f(\langle n, B\rangle) \leq f(\langle n', B'\rangle)$ is true for all $\langle n', B'\rangle \in Y_i^\sharp$ satisfying $B' \cap F \neq \emptyset$. Recall that $f$ is the bijective map which determines the order (i.e., the indices) of attributes in $\mathbb{K}_i^\sharp$.

Moreover, $B \cap F \neq \emptyset$ yields there is $y \in B$ such that $y \in E^{\uparrow_{\mathbb{K}}}$. Hence, $y \in \lfloor E^{\uparrow_{\mathbb{K}_i^\sharp}} \rfloor$ which in turn means that $B \subseteq \lfloor E^{\uparrow_{\mathbb{K}_i^\sharp}} \rfloor$ because all attributes from $B$ are indistinguishable on objects from $E \subseteq X_i^\sharp$. Therefore, $B \subseteq \lfloor E^{\uparrow_{\mathbb{K}_i^\sharp}} \rfloor \subseteq F$. Since $B \subseteq F$ and $\langle n, B\rangle \in Y_i^\sharp$, we have by assumption $n = 0$. Hence, for attribute $\langle n, B\rangle$, Algoritm 1 can proceed to line 4. Let $\langle C, D\rangle$ be defined by $C = \{\langle n, B\rangle\}^{\downarrow_{\mathbb{K}_i^\sharp}}$ and $D = C^{\uparrow_{\mathbb{K}_i^\sharp}}$ which corresponds to calling CLOSURE with $\mathbb{K}_i^\sharp$ and $\langle n, B\rangle$ as its arguments. We now check that the canonicity test succeeds. If $\langle n', B'\rangle \in D$, then clearly $B' \subseteq F$ because $B \subseteq F$ and $\langle x, \langle n', B'\rangle\rangle \in I_i^\sharp$ holds for any $x \in X_i^\sharp$ such that $\langle x, \langle n, B\rangle\rangle \in I_i^\sharp$. Hence, using the assumption, it follows that $n' = 0$. Thus, all flags of attributes from $D$ are zero, i.e. the canonicity test succeeds. As a consequence, we can put $\mathbb{K}_{i+1}^\sharp = \text{REDUCE}(\mathbb{K}_i^\sharp, C, D)$ and we have $\mathbb{K}_i^\sharp \prec \mathbb{K}_{i+1}^\sharp$. Moreover, Lemma 6.2 yields that $\mathbb{K}_{i+1}^\sharp$ is $\mathbb{K}$-representative.

It remains to show that $\mathbb{K}^\sharp_{i+1}$ satisfies the property that if $\langle n, B\rangle \in Y^\sharp_{i+1}$ and $B \subseteq F$, then $n = 0$. Take any $\langle n, B\rangle \in Y^\sharp_{i+1}$ such that $B \subseteq F$. Since $\mathbb{K}^\sharp_{i+1}$ results by reduction and clarification, there are $\langle n_j, B_j\rangle \in Y^\sharp_i$ ($j \in J$) such that $B = \bigcup_{j\in J} B_j$. We have $B_j \subseteq F$ ($j \in J$), i.e., $n_j = 0$. Since $\mathbb{K}^\sharp_{i+1} = \textsc{Reduce}(\mathbb{K}^\sharp_i, C, D)$ for $C = \{\langle n, B\rangle\}^{\downarrow_{\mathbb{K}^\sharp_i}}$ where $\langle n, B\rangle$ was chosen with the least possible index according to $f$, during the reduction, no attribute $\langle n_j, B_j\rangle$ was given a nonzero flag. During the subsequent clarification, some of the attributes $\langle n_j, B_j\rangle$ can be merged together with other attributes with zero flags but they cannot be merged with attributes with nonzero flags (otherwise, it would contradict the fact that $\langle E, F\rangle$ is a formal concept). Therefore, $n = \sum_{j\in J} n_j = 0$, proving the property for $\mathbb{K}^\sharp_{i+1}$.

In order to finish the proof, observe that the sequence can be extended only finitely may times and for $\mathbb{K}^\sharp_i \prec \mathbb{K}^\sharp_{i+1}$, we have $X^\sharp_i \supset X^\sharp_{i+1} \supseteq E$. Hence, after finitely many steps, we obtain $\mathbb{K}^\sharp_n$ with $E = X^\sharp_n$ and thus $F = \textsc{Int}(\mathbb{K}^\sharp_n, Y)$ since $\mathbb{K}^\sharp_n$ if $\mathbb{K}$-representative. □

**Theorem 6.5. (soundness of Algorithm 1)**
During the invocations of COMPUTE, each formal concept in $\mathbb{K}$ is stored exactly once.

**Proof:**
Using Lemma 6.4, each formal concept in $\mathbb{K}$ is stored at least once. Thus, it suffices to prove that each of them is stored at most once. We prove this by showing uniqueness of sequences constructed in the proof of Lemma 6.4. Inspecting the proof of Lemma 6.4, one can see that $\mathbb{K}^\sharp_{i+1}$ is determined from $\mathbb{K}^\sharp_i$ by reduction which uses a formal concept in $\mathbb{K}^\sharp_i$ generated by the least possible attribute $\langle n, B\rangle \in Y^\sharp_i$ such that $B \subseteq F$. If we would have chosen other attribute $\langle n', B'\rangle \in Y^\sharp_i$ such that $B' \subseteq F$ instead of $\langle n, B\rangle$, then $\mathbb{K}^\sharp_{i+1} = \textsc{Reduce}(\mathbb{K}^\sharp_i, C, D)$ for $C = \{\langle n', B'\rangle\}^{\downarrow_{\mathbb{K}^\sharp_i}}$ and $D = C^{\uparrow_{\mathbb{K}^\sharp_i}}$ would contain an attribute $\langle n'', B''\rangle$ such that $B'' \cap F \neq \emptyset$, $B \subseteq B''$, and $n'' > 0$. The attribute $\langle n'', B''\rangle$ would remain in any $R$-context (either directly or being merged with other attributes) that would further extend the sequence. This follows from the fact that once an attribute has a nonzero flag, it is not removed by any reduction from an $R$-context (it can be merged together with other attributes during clarification but the nonzero flag remains). Thus, the selection of $\langle n', B'\rangle \in Y^\sharp_i$ would cause that the sequence $\mathbb{K}^\sharp_1 \prec \cdots \prec \mathbb{K}^\sharp_i \prec \mathbb{K}^\sharp_{i+1}$ cannot be extended to a sequence where the last element is an $R$-context $\mathbb{K}^\sharp_n$ with $X^\sharp_n = E$, meaning that $\langle E, F\rangle$ would not be stored. Altogether, we have shown that for any formal concept the sequence constructed in the proof of Lemma 6.4 is uniquely given. □

## 7. Complexity and Efficiency Issues

In this section, we inspect worst-case complexity of Algorithm 1 and the underlying operations and present experimental evaluation of its performance compared to other algorithms from the CbO family.

The asymptotic worst-case time complexity of Algorithm 1 is the same as in the case of CbO and FCbO, i.e., $O(|\mathcal{B}(X, Y, I)|\cdot|X|\cdot|Y|^2)$. Indeed, for each formal concept, i.e., for each invocation of COMPUTE, one has to determine the reduced and clarified context which is the argument passed to COMPUTE. This can be done as follows: first, one sorts all attributes in an $R$-context according to their support. If the support of two different attributes is the same, the attributes can be additionally sorted lexicographically according to sets of objects having those attributes. This can be done in $O(|X|\cdot|Y|\cdot\log|Y|)$ time. Then, attributes that need to be grouped together during clarification can be identified in a single pass through

the set of attributes and the sets of objects having the attributes, i.e. in $O(|X|\cdot|Y|)$ time. Altogether, the $R$-context is determined in $O(|X|\cdot|Y|\cdot\log|Y|)$ time. Then, Algorithm 1 proceeds as in CbO, i.e., for each attribute, it computes a new closure in $O(|X|\cdot|Y|)$ time and performs the canonicity test in $O(|Y|)$ time. Thus, a single invocation of COMPUTE is done in $O(|X|\cdot|Y|^2)$ time, showing that the asymptotic worst-case time complexity of the algorithm is $O(|\mathcal{B}(X,Y,I)|\cdot|X|\cdot|Y|^2)$. In the case of time delay [10], Algorithm 1 has the same polynomial time delay $O(|Y|^3\cdot|X|)$ as CbO, cf. [17]. The argument remains the same as in the case of CbO.

In order to show the performance of the algorithm compared to other algorithms from the CbO family, we present a set of experiments involving both real-world and artificial datasets and comparison with similar algorithms. All the experiments focus on the total number of computed closures since it is a feature significantly affecting performance of all the algorithms in the CbO family. Table 5 shows counts of closures computed while processing real-world datasets using the CbO, FCbO, and Algorithm 1. Note that the table contains two rows for results of both FCbO and CbO. The rows labeled "ordered" present efficiency of the algorithms if the additional preprocessing step of ordering attributes of input data table according to their support is applied, cf. [13].

From Table 5 it follows that the new algorithm needs to compute considerably less closures than the other algorithms. It seems that this is a general tendency. The tendency is further illustrated by Table 6 and Table 7 containing average counts of computed closures while processing a set of 1,000 artificial data tables. For this experiment we have considered tables of size $50 \times 50$, where density of 1s is 10 % and 33 %, respectively, and 1s are distributed approximately normally among attributes.

Table 5. Number of closures computed by selected algorithms from CbO family

| | debian tags | anon. web. | mushroom | tic-tac-toe |
|---|---|---|---|---|
| size | $14,315 \times 475$ | $32,710 \times 295$ | $8,124 \times 119$ | $958 \times 29$ |
| density | $< 1\,\%$ | $1\,\%$ | $19\,\%$ | $34\,\%$ |
| # concepts | $38,977$ | $129,009$ | $238,710$ | $59,505$ |
| Algorithm 1 | $44,221$ | $135,925$ | $246,181$ | $65,567$ |
| FCbO (ordered) | $298,641$ | $398,147$ | $299,201$ | $89,930$ |
| FCbO | $679,911$ | $1,475,341$ | $426,563$ | $128,434$ |
| CbO (ordered) | $960,106$ | $785,394$ | $1,321,524$ | $185,738$ |
| CbO | $12,045,680$ | $27,949,552$ | $4,006,498$ | $221,608$ |

Table 6. Computed closures in datasets of size $50 \times 50$ with 10 % density of 1s

| | mean value | standard deviation | median value |
|---|---|---|---|
| CbO | $3,359.88$ | $505.51$ | $3294$ |
| CbO (ordered) | $1,394.08$ | $78.19$ | $1,395$ |
| FCbO | $860.41$ | $49.17$ | $860$ |
| FCbO (ordered) | $853.87$ | $47.80$ | $852$ |
| Algorithm 1 | $240.83$ | $8.34$ | $241$ |
| # concepts | $227.58$ | $6.79$ | $228$ |

Table 7. Computed closures in datasets of size $50 \times 50$ with 33 % density of 1s

| | mean value | standard deviation | median value |
|---|---|---|---|
| CbO | $332,253.55$ | $65,135.75$ | $326,097$ |
| CbO (ordered) | $44,074.43$ | $6,345.95$ | $43,975$ |
| FCbO | $43,787.87$ | $6,175.53$ | $43,778$ |
| FCbO (ordered) | $32,059.09$ | $4,350.26$ | $32,057$ |
| Algorithm 1 | $25,754.40$ | $3,565.85$ | $25,776$ |
| # concepts | $24,945.64$ | $3,401.93$ | $24,958$ |

Table 8. Ratios of concepts computed multiple times

| | debian tags | anon. web. | mushroom | tic-tac-toe |
|---|---|---|---|---|
| size | $14,315 \times 475$ | $32,710 \times 295$ | $8,124 \times 119$ | $958 \times 29$ |
| density | $< 1\,\%$ | $1\,\%$ | $19\,\%$ | $34\,\%$ |
| Algorithm 1 | $0.13$ | $0.05$ | $0.03$ | $0.10$ |
| FCbO (ordered) | $6.66$ | $2.08$ | $0.25$ | $0.51$ |
| FCbO | $16.44$ | $10.43$ | $0.78$ | $1.15$ |
| CbO (ordered) | $23.63$ | $5.08$ | $4.53$ | $2.12$ |
| CbO | $308.04$ | $215.64$ | $15.78$ | $2.72$ |

Apparently, the new method of computing formal concepts can reduce the total number of computed closures by several orders of magnitude. The factor of improvement depends on many aspects, especially the size of input data. To reduce the influence of this aspect while evaluating algorithms, we use the ratio of concepts computed multiple times (i.e., redundant concepts) to the total number of concepts present in the dataset. Table 8 depicts such ratios for previously discussed real-world datasets. As one can see, the new algorithm while processing *mushroom* dataset computes only 3 % of concepts multiple times. This strongly contrasts with CbO which computes more than fifteen times more concepts than necessary. Furthermore, in case of large and sparse datasets like *anonymous web* and *debian tags* the new algorithm needs to compute only a small fraction of concepts multiple times. This is also a remarkable contrast with the other algorithms since, for instance, CbO computes even hundreds of times more concepts than Algorithm 1.

These tendencies are quite general. For instance, Figure 3 depicts ratios of concepts computed multiple times and their relationship to the number of attributes in the formal context. In this experiment, we have used multiple randomly generated formal contexts having 1,000 objects and various counts of attributes. We have considered data tables with density 5 % and approximately normal distribution of 1s among attributes. Interestingly, it seems that the number of objects has no noticeable impact on the efficiency in terms of concepts computed multiple times as it is shown, e.g., in Figure 4. This figure presents efficiency of algorithms in relationship to the number of objects. In this experiment we have also used artificial datasets and each data table had 100 attributes, various counts of objects, and 1s were distributed approximately normally among attributes with 5 % density. Note that, since CbO (without the preprocessing step) shows a very poor performance, it has been omitted from the chart for the sake of readability.

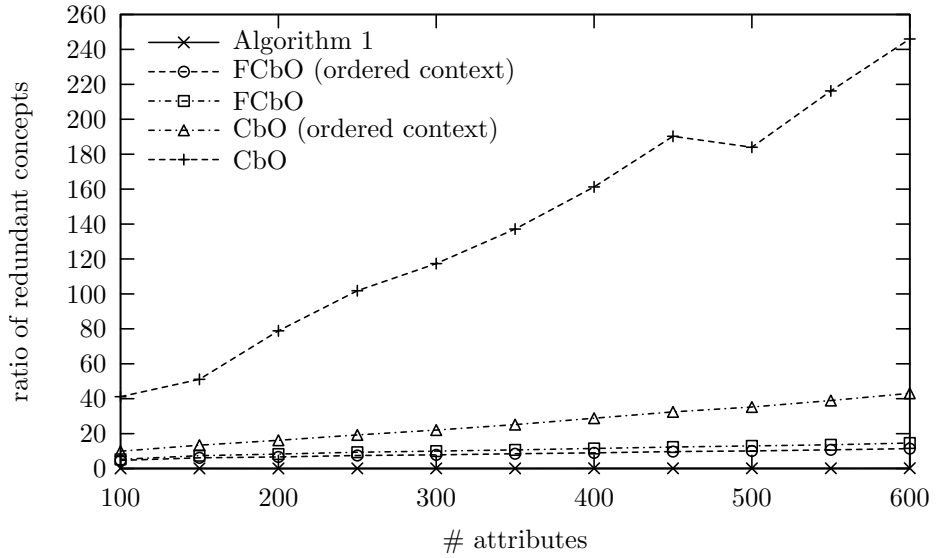


Figure 3. Ratios of concepts computed multiple times and their relationship to the number of attributes

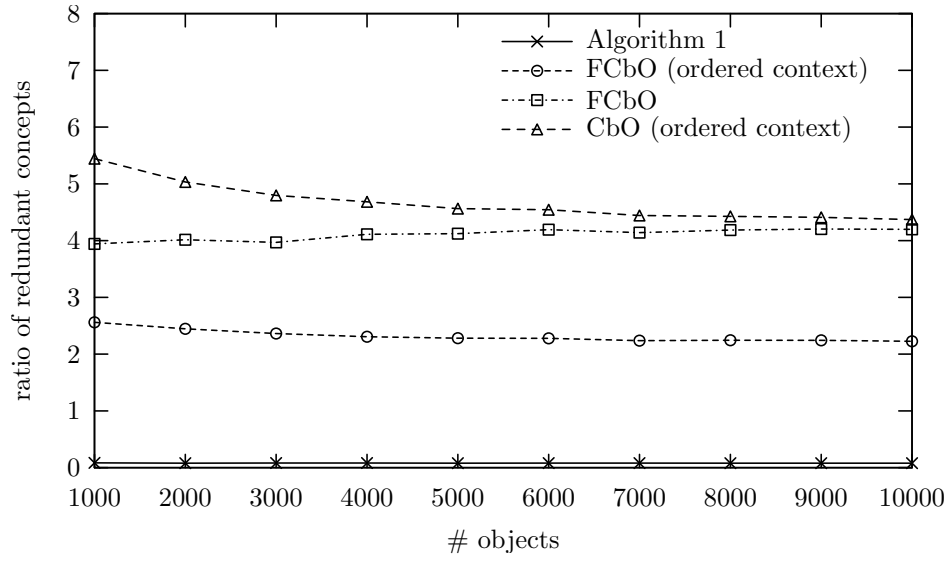


Figure 4. Ratios of concepts computed multiple times and their relationship to the number of objects

## References


[1] Agrawal R., Imielinski T., Swami A. N.: Mining association rules between sets of items in large databases, *Proc. ACM Int. Conf. of Management of Data*, 1993, 207–216.

[2] Andrews S.: In-Close, a Fast Algorithm for Computing Formal Concepts, *Supplementary Proceedings of ICCS '09* (S. Rudolph, F. Dau, S. O. Kuznetsov, Eds.), CEUR WS, vol. 483, 2009.

[3] Belohlavek R., Vychodil V.: Discovery of optimal factors in binary data via a novel method of matrix decomposition, *J. Comput. Syst. Sci.*, **76**(1), 2010, 3–20.

[4] Carpineto C., Romano G.: *Concept data analysis. Theory and applications*, J. Wiley, 2004.

[5] Hereth Correia J., Stumme G., Wille R., Wille U.: Conceptual knowledge discovery—a human-centered approach, *Applied Artificial Intelligence*, **17**(3), 2003, 281–302.

[6] Ganter B.: Two basic algorithms in concept analysis, *Proc. ICFCA 2010*, LNCS 5986, 2010, 312–340 (reprint of Technical Report FB4-Preprint No. 831, TH Darmstadt, 1984).

[7] Ganter B., Wille R.: *Formal concept analysis. Mathematical foundations*, Springer, Berlin, 1999.

[8] Goldberg L. A.: *Efficient Algorithms for Listing Combinatorial Structures*, Cambridge University Press, 1993.

[9] Hettich S., Bay S. D.: *The UCI KDD Archive*, University of California, Irvine, School of Information and Computer Sciences, 1999.

[10] Johnson D. S, Yannakakis M., Papadimitriou C. H.: On generating all maximal independent sets, *Information Processing Letters*, **27**(3), 1988, 119–123.

[11] Koester B.: *FooCA – Web Information Retrieval with Formal Concept Analysis*, Verlag Allgemeine Wissenschaft, 2006.

[12] Krajca P., Outrata J., Vychodil V.: Parallel algorithm for computing fixpoints of Galois connections, *Ann. Math. Artif. Intell.*, **59**(2), 2010, 257–272.

[13] Krajca P., Outrata J., Vychodil V.: Advances in algorithms based on CbO, *Proc. CLA 2010* (M. Kryszkiewicz, S. Obiedkov, Eds.), CEUR WS, vol. 672, 2010, 325–337 (`http://ceur-ws.org/Vol-672/paper29.pdf`).

[14] Kuznetsov S. O.: Interpretation on graphs and complexity characteristics of a search for specific patterns, *Automatic Documentation and Mathematical Linguistics*, **24**(1), 1989, 37–45.

[15] Kuznetsov S. O.: A fast algorithm for computing all intersections of objects in a finite semi-lattice (Быстрый алгоритм построения всех пересечений объектов из конечной полурешетки, in Russian), *Automatic Documentation and Mathematical Linguistics*, **27**(5), 1993, 11–21.

[16] Kuznetsov S. O.: Learning of Simple Conceptual Graphs from Positive and Negative Examples. *Proc. PKDD*, 1999, 384–391.

[17] Kuznetsov S. O., Obiedkov S. A.: Comparing performance of algorithms for generating concept lattices, *J. Exp. Theor. Artif. Int.*, **14**, 2002, 189–216.

[18] Kneale W., Kneale M.: *The Development of Logic*, Oxford University Press, USA, 1985.

[19] Lindig C.: Fast concept analysis. *Working with Conceptual Structures–Contributions to ICCS*, 2000, 152–161.

[20] van der Merwe D., Obiedkov S. A., Kourie D. G.: AddIntent: A New Incremental Algorithm for Constructing Concept Lattices. *Proc. ICFCA 2004*, LNAI 2961, 2004, 205–206.

[21] Miettinen P., Mielikäinen T., Gionis A., Das G., Mannila H.: The discrete basis problem. *Proc. PKDD*, 2006, 335–346.

[22] Norris E. M.: An Algorithm for Computing the Maximal Rectangles in a Binary Relation, *Revue Roumaine de Mathématiques Pures et Appliquées*, **23**(2), 1978, 243–250.

[23] Outrata J., Vychodil V.: Fast algorithm for computing fixpoints of Galois connections induced by object-attribute relational data, *Inf. Sci.*, doi:10.1016/j.ins.2011.09.023.

[24] Pasquier N., Bastide Y., Taouil R., Lakhal L.: Efficient mining of association rules using closed itemset lattices, *Inf. Syst.*, **24**(1), 1999, 25–46.

[25] Snelting G., Tip F.: Reengineering class hierarchies using concept analysis, *ACM Transactions on Programming Languages and Systems*, **22**(3), 2000, 540–582.

[26] Tonella P.: Using a concept lattice of decomposition slices for program understanding and impact analysis, *IEEE Transactions on Software Engineering*, **29**(6), 2003, 495–509.

[27] Wille R.: Restructuring lattice theory: an approach based on hierarchies of concepts. *Ordered Sets*, 1982, 445–470, Dordrecht-Boston.

[28] Zaki M. J., Hsiao C.-J.: CHARM: An Efficient Algorithm for Closed Itemset Mining. *Proc. SIAM DM*, 2002.

[29] Zaki M. J.: Mining non-redundant association rules, *Data Mining and Knowledge Discovery*, **9**, 2004, 223–248.