The Closed Keys Base of Frequent Itemsets

Viet Phan Luong

Laboratoire d’Informatique Fondamentale de Marseille, LIF-UMR CNRS 6166
C.M.I. de l’Université de Provence
39, rue F. Joliot Curie, 13453 Cedex 13, France
phan@gyptis.univ-mrs.fr

Abstract. In data mining, concise representations are useful and necessary to apprehending voluminous results of data processing. Recently many different concise representations of frequent itemsets have been investigated. In this paper, we present yet another concise representation of frequent itemsets, called the closed keys representation, with the following characteristics: (i) it allows to determine if an itemset is frequent, and if so, the support of the itemset is immediate, and (ii) basing on the closed keys representation, it is straightforward to determine all frequent key itemsets and all frequent closed itemsets. An efficient algorithm for computing the closed key representation is offered. We show that our approach has many advantages over the existing approaches, in terms of efficiency, conciseness and information inferences.

1 Introduction

Searching for frequent itemsets is a primary step in mining association rules, episode rules, sequential patterns, clusters, etc. The number of frequent itemsets in a real application can be exponential with respect to the number of items considered in the application. This is a problem not only to online data mining, but also to end-users to apprehending the mining results. A solution of this problem is the concise representations of frequent itemsets and association rules [3, 7, 10, 12, 13, 15]. Existing approaches to the concise representations of frequent itemsets are the representations based on closed itemsets [9, 15], key (generator) itemsets [4], disjunction-free itemsets [5], and disjunction-free generator itemsets [8]. Each of these representations is lossless: it allows to determine if an itemset is frequent, and if so, the support of the itemset can be computed from the supports of the itemsets in the representation. However, as we shall see such a computation of the support is not straightforward. Moreover, in the applications of the concise representations of frequent itemsets, as the representations of association rules, we need to know both closed itemsets and key itemsets [7, 12, 13]. Searching for the key itemsets from the closed itemsets representation or the disjunction-free itemsets representation is not straightforward. It is the same to searching for the closed itemsets from the generators representation or the disjunction-free generators representation.

In this paper, we present yet another concise representation of frequent itemsets, that we call the closed keys representation, with the following properties:
(i) it is lossless, and (ii) basing on the representation, it is straightforward to
determine the supports of all frequent itemsets, and to find all frequent key itemsets
and all closed itemsets. An efficient algorithm for finding the representation
is offered. Comparisons with related work, in terms of efficiency, conciseness
and informational inferences, are discussed. We show that our approach has many
advantages over the existing approaches.

The paper is organized as follows. In Section 2, we remind the main concepts
on frequent itemsets, closed itemsets and key itemsets. Properties concerning the
inference of supports of itemsets are presented in Section 3. The concept of the
closed keys representation is given at the end of Section 3. Section 4 is devoted to
the algorithm for searching the closed keys representation. The correctness and
completeness of the algorithm are proved. Related work is discussed in Section
5, which ends with remarks and conclusions.

2 Preliminaries

We consider a dataset which is a triple $\mathcal{D} = (\mathcal{O}, \mathcal{I}, \mathcal{R})$, where $\mathcal{O}, \mathcal{I}, \mathcal{R}$ are finite
non-empty sets. An element in $\mathcal{I}$ is called an item, an element in $\mathcal{O}$ is called an
object or a transaction. $\mathcal{R}$ is a binary relation on $\mathcal{O}$ and $\mathcal{I}$. A couple $(o, i) \in \mathcal{R}$
represents the fact that the object (transaction) $o$ has the item $i$. A subset $I \subseteq \mathcal{I}$
is called an itemset of $\mathcal{D}$. An itemset consisting of $k$ items is called a $k$-itemset.
The Galois connection [6] between $2^\mathcal{O}$ and $2^\mathcal{I}$ is a couple of functions $(g, f)$ where
$g(I) = \{o \in \mathcal{O} \mid \forall i \in I, (o, i) \in \mathcal{R}\}$, and $f(O) = \{i \in \mathcal{I} \mid \forall o \in \mathcal{O}, (o, i) \in \mathcal{R}\}$. Intuitively, $g(I)$ is the set of all objects in $\mathcal{O}$ that share in common the items in
$I$, and dually, $f(O)$ is the set of all items that the objects in $O$ share in common.
The functions $g$ and $f$ can be extended with $g(\emptyset) = \emptyset$ and $f(\emptyset) = \emptyset$. The function
$g$ is antimonotonic: for all $I_1, I_2 \subseteq \mathcal{I}$, if $I_1 \subseteq I_2$, then $g(I_2) \subseteq g(I_1)$. The function
$f$ is also antimonotonic. The Galois closure operators are the following functions:
$h = f \circ g$ and $h' = g \circ f$, where $o$ denotes the composition of functions. $h$ and
$h'$ are monotonic. Given an itemset $I$, $h(I) = f(g(I))$ is called the closure of $I$.
Indeed, for all $I \in \mathcal{I}$, $I \subseteq h(I)$ (Extension) and $h(h(I)) = h(I)$ (Idempotency).
An itemset $I$ is said to be closed if $I = h(I)$, An itemset $I$ is called a key itemset
or a generator if for every itemset $I' \subseteq I$, $h(I') = h(I)$ implies $I' = I$. That is
$I$ is a key itemset if there is no itemset $I'$ strictly included in $I$ and having the same
closure as $I$.

The support of $I$, denoted by $sup(I)$, is $sup(I) = \text{card}(g(I))/\text{card}(\mathcal{O})$, where
$\text{card}(X)$ denotes the cardinality of $X, \forall X \subseteq \mathcal{O}$. Given a support threshold,
denoted by $\text{minsup}$, $0 \leq \text{minsup} \leq 1$, an itemset $I$ is said to be frequent if
$sup(I) \geq \text{minsup}$. An itemset $I$ is called a frequent key (or closed) itemset if $I$ is
frequent and $I$ is a key (respectively closed) itemset.

An association rule [1] is an expression of the form $I_1 \Rightarrow I_2$, where $I_1, I_2 \subseteq \mathcal{I}$.
Let $r$ be an association rule, denoted by $r : I_1 \Rightarrow I_2$. Its support and confidence,
denoted by $sup(r)$ and $conf(r)$ respectively, are $sup(r) = sup(I_1 \cup I_2)$, and
$conf(r) = supp(r)/sup(I_1)$. A general form of association rules, called disjunctive
association rules [5], is $r : I \Rightarrow I_1 \lor I_2$, where $I \subseteq \mathcal{I}$, with support and
confidence defined by $sup(r) = sup(I \cup I_1) + sup(I \cup I_2) - sup(I \cup I_1 \cup I_2)$, and $conf(r) = sup(r)/sup(I)$, let $r$ be an association rule (normal or disjunctive), $r$ is said to be exact (certain) if $conf(r) = 1$; otherwise, $r$ is said to be approximate.

Example 1. Consider a dataset represented in Table a. The itemsets of the dataset are classified with respect to their supports in Table b.

<table>
<thead>
<tr>
<th>Table a</th>
<th>Table b</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OID</strong></td>
<td><strong>Items</strong></td>
</tr>
<tr>
<td>1</td>
<td>A - D</td>
</tr>
<tr>
<td>2</td>
<td>B - E</td>
</tr>
<tr>
<td>3</td>
<td>A B C</td>
</tr>
<tr>
<td>4</td>
<td>A - C</td>
</tr>
<tr>
<td>5</td>
<td>A B C</td>
</tr>
</tbody>
</table>

In short an itemset is denoted by juxtaposition of its items, so is the union of itemsets, when no confusion is possible. For example, $ACE$ denotes the itemset $\{A, C, E\}$, and $I_1I_2$ denotes $I_1 \cup I_1$, where $I_1, I_2$ are itemsets.

Let $\text{minsup} = 2/5$. The frequent closed itemsets of $D$ are $ABCE$, $ACE$, $BE$, $A$, $E$, $\emptyset$. The frequent itemsets of $D$ are $AB, AE, BC, A, B, C, E, \emptyset$. The association rule $B \rightarrow ACE$ is approximate. The association rule $B \rightarrow C \vee E$ is disjunctive and exact.

3. Closed keys representation

We start this section by providing the basis for support inferences. Then we define the concept of the closed keys representation, and show how frequent itemsets can be determined on the representation. Some results are similar to the results in [4]. However, we shall show how our results are distinct from those in [4].

Lemma 1. Let $I, J$ be itemsets. Then $g(I \cup J) = g(I) \cap g(J)$.

Proposition 1. Let $I_1, I_2$ be itemsets such that $I_1 \subset I_2$. If $g(I_1) = g(I_2)$ then for every $I \supset I_2$, $g(I) = g(I_1 \cup (I - I_2))$.

Proof. Suppose that $I_1 \subset I_2$ and $g(I_1) = g(I_2)$. Let $I$ be an itemset such that $I \supset I_2$. $I$ can be represented as $I = I_2 \cup (I_1 \cup (I - I_2))$. Therefore, by Lemma 1, $g(I) = g(I_2) \cap g(I_1 \cup (I - I_2))$. As $g(I_2) = g(I_1)$, we have $g(I) = g(I_1) \cap g(I_1 \cup (I - I_2))$. Now, as $g(I_1 \cup (I - I_2)) \subseteq g(I_1)$, we have: $g(I) = g(I_1 \cup (I - I_2))$.

In [4], it was shown that $g(I) = g(I - (I_2 - I_1))$. In fact, $I_1 \cup (I - I_2) = I - (I_2 - I_1)$. However, $I_1 \cup (I - I_2)$ is computationally more efficient than $I - (I_2 - I_1)$, because under the condition $I_1 \subset I_2 \subset I$, we have $I_1 \cap (I - I_2) = \emptyset$, so the union of $I_1$ and $I - I_2$ can be implemented by just insertion.

The following corollaries are direct consequences of Proposition 1.
Corollary 1. Let \( I_1, I_2 \) be itemsets such that \( I_1 \subset I_2 \). If \( sup(I_1) = sup(I_2) \) then for every \( I \supset I_2 \), \( sup(I) = sup(I_1 \cup (I - I_2)) \).

Corollary 2. Let \( I \) be an itemset. If \( I \) is not a key itemset then for every \( J \supset I \), \( J \) is not a key itemset.

After Corollary 2, if we are only interested in key itemsets, then in an incremental method for generating key itemsets, we can discard non-key itemsets.

Definition 1. An itemset \( K \) is called a maximal key itemset included in an itemset \( X \) if \( K \) is a key itemset and there is no key itemset \( K' \) such that \( K \subset K' \subset X \).

In the above definition, \( K \) is required to be a key itemset, but \( sup(K) \) is not necessarily equal to \( sup(X) \). However, the following property holds.

Lemma 2. If \( X \) is a non-key itemset, then any key itemset \( K \) included in \( X \), such that \( sup(X) = sup(K) \), is a maximal key itemset included in \( X \).

Proposition 2. Let \( X \) be an itemset. Let \( K_1, K_2, \ldots, K_m \) be maximal key itemsets included in \( X \). Then \( sup(X) = \min \{ sup(K_i) \mid K_i, i = 1 \ldots m \} \)

Proof. We have \( sup(X) \leq \min \{ sup(K_i) \mid K_i, i = 1 \ldots m \} \). If \( X \) is a key itemset, then clearly \( sup(X) = \min \{ sup(K_i) \mid K_i, i = 1 \ldots m \} \), where \( m = 1 \). Otherwise, \( X \) is not a key. Then there exists a key itemset \( K \) such that \( K \subset X \) and \( sup(K) = sup(X) \). Such a key itemset \( K \) is a maximal key itemset included in \( X \) (Lemma 2). Hence, \( sup(X) = sup(K) \geq \min \{ sup(K_i) \mid K_i, i = 1 \ldots m \} \). Thus, \( sup(X) = \min \{ sup(K_i) \mid K_i, i = 1 \ldots m \} \).

In [4], it was shown that if \( X \) is a non-key \( k \)-itemsets, \( k \geq 2 \), then the support of \( X \) is \( sup(X) = \min \{ sup(I_{k-1}) \mid I_{k-1} \subset X \} \). Proposition 2 is distinct from this result on two points (i) Proposition 2 is valid not only for non-key itemsets, but also for key itemsets, and (ii) in order to compute the support of \( X \), we do not need to consider all \( (k-1) \)-itemsets included in \( X \), but only the maximal key itemsets included in \( X \).

Proposition 2 means that if we know the supports of all key itemsets, then we can infer the support of any other itemset. However, if we are only interested in determining the frequent key itemsets with their supports, then we can base on only frequent key itemsets.

Theorem 1. If there exists a frequent key itemset \( K \) such that \( K \subset X \) and \( X \subset h(K) \), then \( sup(X) = sup(K) \). Otherwise, \( X \) is not frequent.

Proof. If there exists a frequent key itemset \( K \) such that \( K \subset X \) and \( X \subset h(K) \), then \( sup(X) \leq sup(K) \) and \( sup(h(K)) \leq sup(X) \). As \( sup(K) = sup(h(K)) \), we have \( sup(X) = sup(K) = sup(h(K)) \). Otherwise, by contradiction suppose that \( X \) is frequent. We have either \( X \) is a key or \( X \) is not a key. In the former case, the contradiction is immediate, because \( X \) is a frequent key itemset included in itself. In the latter case, there exists a key itemset \( K \subset X \) such that \( sup(K) = sup(X) \). Therefore \( g(K) = g(X) \), and then \( h(K) = f(g(K)) = f(g(X)) = h(X) \). Hence, \( X \subset h(X) = h(K) \). As \( X \) is frequent, \( K \) is frequent. Thus, contradiction. \( \Box \).
Definition 2. Let $D$ be a dataset. The closed keys representation of $D$, with respect to a support threshold $\text{minsup}$, denoted by $\mathcal{H}(D, \text{minsup})$, is the set
$$(\{K, h(K) - K, \text{sup}(K)\} | K \text{ is a key itemset of } D \text{ and } \text{sup}(K) \geq \text{minsup}).$$

By Theorem 1, basing on the closed keys representation, it is straightforward to determine if an itemset $I$ is frequent, and if so, its support already available, without any computation.

Example 1 (continued). The closed keys representation of the dataset in Table a, with respect to $\text{minsup} = 2/5$, consists of $(\emptyset, 0, 1), (A, \emptyset, 4/5), (B, E, 3/5), (C, AE, 3/5), (E, \emptyset, 4/5), (AB, CE, 2/5), (AE, C, 3/5), (BC, AE, 2/5)$.

4 Generating the closed keys representation

We propose an incremental algorithm as Apriori [2] to generate the closed keys representation. We start with the empty itemset, which is of course a key, with support 1. Then we compute the supports of all 1-itemsets. For each 1-itemset $I$, if $\text{sup}(I) = 1$, then $I$ is a non-key itemset. The remaining 1-itemsets are key itemsets. Let $Y_0$ be the union of all non-key 1-itemsets. Clearly, $Y_0$ is the closure of the empty itemset. A triple $(\emptyset, Y_0, 1)$ is added to the closed keys representation. In step $i \geq 2$, we consider the $i$-itemsets built on the frequent $(i-1)$-itemsets, until the set of key $(i-1)$-itemsets is empty.

4.1 Generating key itemset candidates

Let $X$ and $Y$ be frequent $(i-1)$-itemsets such that $X - Y$ and $Y - X$ are singletons and $X - Y > Y - X$ in lexicographic order. Then $XY$ is an $i$-itemset candidate. We have the following cases:

(a) $X$ is not key. Then $XY$ is not key (Corollary 2). As $X$ is frequent, there exists a frequent $(i-2)$-itemset $Z$ such that $Z \subseteq X$ and $\text{sup}(Z) = \text{sup}(X)$. By Corollary 1, $\text{sup}(XY) = \text{sup}(Z \cup (XY - X)) = \text{sup}(Z \cup (Y - X))$. As $Z$ is an $(i-2)$-itemset and $(Y - X)$ is disjoint from $Z$, we have $Z \cup (Y - X)$ is an $(i-1)$-itemset. Therefore, we search $(Z \cup (XY - X))$ among the frequent $(i-1)$-itemsets. If it is there, then $XY$ is frequent, and the itemset $Z \cup (Y - X)$ is linked to $XY$, for use in the next step. Otherwise, $XY$ is not frequent. The case where $Y$ is not key is similar.

(b) Both $X$ and $Y$ are keys. There are two subcases:

(b.1) There exists a frequent non-key $(i-1)$-itemset $Z$ such that $Z \subseteq XY$. In this case, $XY$ is not a key. As $Z$ is a frequent non-key itemset, there exists a frequent $(i-2)$-itemset $T$ such that $T \subseteq Z$ and $\text{sup}(T) = \text{sup}(Z)$, and $\text{sup}(XY) = \text{sup}(T \cup (XY - Z))$. Similarly to case (a), by searching among the frequent $(i-1)$-itemsets previously discovered, we can see if $XY$ is frequent, and if so, the itemset $T \cup (XY - Z)$ is linked to $XY$, for use in the next step. Otherwise, $XY$ is infrequent.

(b.2) Else, if all $(i-1)$-itemsets which are subsets of $XY$ are frequent then $XY$ is a candidate. Otherwise, $XY$ is deleted from the list of candidates.
The frequent \( i \)-itemsets generated in points (a) and (b.1) are not key itemsets. They are stored in a list denoted by \( F_i \). The frequent \( i \)-itemsets generated in point (b.2) are stored in a list denoted by \( C_i \).

Notations: For each \( i \)-itemset \( I \), there are associated the following fields:
- \( I \)-key: to indicate if \( I \) is a key itemset, and
- \( I \)-prev: to represent an \( (i-1) \)-itemset \( I_{i-1} \) such that \( I_{i-1} \subseteq I \) and \( \text{sup}(I) = \text{sup}(I_{i-1}) \), if such an itemset exists.

Algorithm GenCandidate

Input: \( F_{i-1}, C_{i-1} \): lists of frequent \((i-1)\)-itemsets.
Output: \( F_i \): list of frequent \( i \)-itemsets, and \( C_i \) list of \( i \)-itemset candidates.
Method:
\( F_i = \emptyset \); \( C_i = \emptyset \);

For each pair of itemsets \((X,Y)\), where \((X, \text{sup}(X)), (Y, \text{sup}(Y)) \in F_{i-1} \cup C_{i-1}\), such that \( X \cap Y \) and \( X \cap Y \) are singletons and \( X \cap Y > X \cap Y \) in lexicographic order, do

If not \( X \)-key then
\[ Z = X \text{prev}; \] // \( Z \) is an \((i-2)\)-itemset: \( Z \subseteq X \) and \( \text{sup}(Z) = \text{sup}(X) \);
If \( Z \cup (Y - X) \) is a frequent \((i-1)\)-itemset then begin
\( XY \)-key = false; \( \text{sup}(XY) = \text{sup}(Z \cup (Y - X)) \);
\( XY \)-prev = \( Z \cup (Y - X) \); insert \((XY, \text{sup}(XY))\) in \( F_i \)
end End

Else If not \( Y \)-key then Begin
\[ Z = Y \text{prev}; \] // \( Z \) is an \((i-2)\)-itemset: \( Z \subseteq Y \) and \( \text{sup}(Z) = \text{sup}(Y) \);
If \( Z \cup (X - Y) \) is a frequent \((i-1)\)-itemset then begin
\( XY \)-key = false; \( \text{sup}(XY) = \text{sup}(Z \cup (X - Y)) \);
\( XY \)-prev = \( Z \cup (X - Y) \); insert \((XY, \text{sup}(XY)) \) in \( F_i \)
end End

Else If there exists a frequent non-key \((i-1)\)-itemset \( Z \) such that \( Z \subseteq XY \) then Begin
\[ T = Z \text{prev}; \] // \( T \) is an \((i-2)\)-itemset: \( T \subseteq Z \) and \( \text{sup}(T) = \text{sup}(Z) \);
If \( T \cup (XY - Z) \) is a frequent \((i-1)\)-itemset then begin
\( XY \)-key = false; \( \text{sup}(XY) = \text{sup}(T \cup (XY - Z)) \);
\( XY \)-prev = \( T \cup (XY - Z) \); insert \((XY, \text{sup}(XY)) \) in \( F_i \)
end End

Else If all \((i-1)\)-itemsets which are subsets of \( XY \) are frequent then begin \( \text{sup}(XY) = 0 \); insert \((XY, \text{sup}(XY)) \) into \( C_i \) end;

4.2 Generating the closed keys representation

In step \( i \geq 2 \), GenCandidate is called with inputs \( F_{i-1} \) and \( C_{i-1} \). It results in \( F_i \) and \( C_i \). If \( C_i \) is not empty, then we access the dataset to compute the supports of the itemsets in \( C_i \), and keep only those which are frequent. Next, we consider each frequent key itemset \( I_{i-1} \). For each \( I_i \in C_i \cup F_i \), if \( \text{sup}(I_i) = \text{sup}(I_{i-1}) \), then we add \( I_i - I_{i-1} \) to \( Y_{i-1} \), which is initially set to empty for each frequent key itemset \( I_{i-1} \), and we set \( I_i \)-prev = \( I_{i-1} \), if \( I_i \in C_i \). In such a case clearly \( I_i \).
is not a key itemset. We shall show that the union of \( I_{i+1} \) and the final value of \( Y_i \), when all \( I_i \in C_i \cup F_i \) are already considered, is the closure of \( I_{i+1} \).

**Algorithm FClosedKeys**

**Input:** A dataset \( \mathcal{D} = (\mathcal{O}, \mathcal{I}, \mathcal{R}) \), and a support threshold \( \text{minsup} \).

**Output:** The closed keys representation of \( \mathcal{D} \), with respect to \( \text{minsup} \).

**Method:**

\[ \emptyset.\text{key} = \text{true}; \quad F_0 = \{ (\emptyset, 1) \}; \quad Y_0 = \emptyset; \quad K_1 = \emptyset; \]

Read the dataset to compute the support of each 1-itemset;

\[ F_1 = \{ (I_1, \text{sup}(I_1)) \mid \text{sup}(I_1) \geq \text{minsup} \}; \]

For each \( (I_1, \text{sup}(I_1)) \in F_1 \) do

- If \( \text{sup}(I_1) = 1 \) then begin \( Y_0 = Y_0 \cup I_1; \quad I_1.\text{prev} = \emptyset; \quad I_1.\text{key} = \text{false} \) end else begin \( I_1.\text{key} = \text{true}; \quad K_1 = K_1 \cup \{ (I_1, \text{sup}(I_1)) \} \) end;

\[ \mathcal{H}_0 = \{ (\emptyset, Y_0, I_1) \}; \quad i = 2; \quad \mathcal{H} = \mathcal{H}_0; \quad \mathcal{H}_{i-1} = \emptyset; \quad C_i = \emptyset; \]

While \( K_{i-1} \neq \emptyset \) do begin

- GenCandidate\( (F_i, C_{i-1}) \); // Results in \( F_i \) and \( C_i \).

  - If \( C_i \neq \emptyset \) then begin

    - Read the dataset to compute the support of each \( i \)-itemset in \( C_i \);

    - Delete from \( C_i \) all itemsets \( I_i \) such that \( \text{sup}(I_i) < \text{minsup} \);

    - For each \( (I_i, \text{sup}(I_i)) \in C_i \) do \( I_i.\text{key} = \text{true} \); // by default

  end;

- For each \( (I_i, \text{sup}(I_i)) \in K_{i-1} \) do begin

  - \( Y_i = \emptyset; \)

  - For each \( (I_i, \text{sup}(I_i)) \in C_i \) do

    - If \( I_i \subseteq I_1 \) and \( \text{sup}(I_i) = \text{sup}(I_1) \) then begin

      - \( Y_i = Y_i \cup (I_i - I_i); \quad I_i.\text{prev} = I_i_1; \quad I_i.\text{key} = \text{false} \)

    end;

  - For each \( (I_i, \text{sup}(I_i)) \in F_i \) do

    - If \( I_i \subseteq I_1 \) and \( \text{sup}(I_i) = \text{sup}(I_1) \) then \( Y_i = Y_i \cup (I_i - I_i) \);

    - \( \mathcal{H}_i = \mathcal{H}_i \cup \{ (I_1, Y_i, \text{sup}(I_i)) \} \);

  end;

- \( \mathcal{H} = \mathcal{H} \cup \mathcal{H}_i \);

- \( K_i = \{ (I_i, \text{sup}(I_i)) \in C_i \mid I_i.\text{key} = \text{true} \} \);

- \( i = i + 1; \quad \mathcal{H}_{i-1} = \emptyset; \)

- Return\( (\mathcal{H}) \).

**Example 1 (continued).** Computing the closed keys representation of the dataset in Table a, with respect to \( \text{minsup} = 2/5 \).

\( K_1 = F_1 = \{ (A, 4/5), (B, 3/5), (C, 3/5), (D, 1/5), (E, 4/5) \}; \)

\( \mathcal{H}_0 = \{ (\emptyset, \emptyset, 1) \}; \quad C_1 = \emptyset. \)

Call GenCandidate\( (F_1, C_1) \), and compute the supports of the itemsets in \( C_2 \):

\( F_2 = \emptyset; \quad C_2 = \{ (AB, 2/5), (AC, 3/5), (AE, 3/5), (BC, 2/5), (BE, 3/5), (CE, 3/5) \}. \)

For each key itemset in \( K_1 \): For \( A \), \( Y_1 = \emptyset \). For \( B \), \( Y_1 = E \), \( BE.\text{prev} = B \), and \( BE \) is not a key itemset. For \( C \), \( Y_1 = AE \), \( AC.\text{prev} = C \), \( AE.\text{prev} = C \), and \( AC \) and \( CE \) are not key itemsets. For \( E \), \( Y_1 = \emptyset \). Hence,

\( \mathcal{H}_1 = \{ (A, \emptyset, 4/5), (B, E, 3/5), (C, AE, 3/5), (E, \emptyset, 4/5) \}, \)

and \( K_2 = \{ (AB, 2/5), (AE, 3/5), (BC, 2/5) \}. \)
Call \textit{GenCandidate}(F_2, C_2):
\[
F_3 = \{(ABC, 2/5), (ABE, 2/5), (ACE, 3/5), (BCE, 2/5)\}; \quad C_3 = \emptyset;
\]
For each key itemset in \(K_2\), if \(AB, Y_2 = CE\), and \(ABC\) and \(ABE\) are not key itemsets. For \(AE, Y_2 = C\), and \(ACE\) is not a key itemset. For \(BC, Y_2 = AE\), and \(ABC\) and \(BCE\) are not key itemsets. Hence,
\[
\mathcal{H}_2 = \{(AB, CE, 2/5), (AE, C, 3/5), (BC, AE, 2/5)\}.
\]
As \(C_3 = \emptyset\), \(K_3\) is also empty, the algorithm stops. The closed keys representation of the dataset \(D\), with respect to \(\text{min}\sup = 2/5\), is
\[
\mathcal{H} = \{(\emptyset, \emptyset, 1), (A, \emptyset, 4/5), (B, E, 3/5), (C, AE, 3/5), (E, \emptyset, 4/5), (AB, CE, 2/5), (AE, C, 3/5), (BC, AE, 2/5)\}.
\]

\textbf{Proposition 3.} In Algorithm \textit{FClosedKeys}, for an itemset \(I_i \in C_i\), if \(I_i\) is not set to be false, then \(I_i\) is a frequent key itemset.

\textbf{Proposition 4.} \textit{FClosedKeys} finds all frequent key itemsets.

\textbf{Proposition 5.} For each triple \((I_{i-1}, Y_{i-1}, \text{sup}(I_{i-1})) \in \mathcal{H}_{i-1}\), \(I_{i-1}\) is a frequent key itemset and \(h(I_{i-1}) = I_{i-1} \cup Y_{i-1}\).

\textit{Proof.} By Proposition 3, \(I_{i-1}\) is a frequent key itemset. For each \(A \in h(I_{i-1}) - I_{i-1}\), \(I_{i-1} \cup \{A\}\) is a frequent \(i\)-itemset, which is either in \(F_i\) or in \(C_i\). For each frequent key \((i-1)\)-itemset \(I_{i-1}\), \textit{FClosedKeys} considers all \(i\)-itemsets \(I_i\) in \(C_i\) and \(F_i\). If \(I_{i-1} \subseteq I_i\) and \(\text{sup}(I_{i-1}) = \text{sup}(I_i)\), then set \(Y_{i-1} = Y_{i-1} \cup (I_i - I_{i-1})\), where \(Y_{i-1}\) is initially set to empty. Thus, \(h(I_{i-1}) = I_{i-1} \cup Y_{i-1}\).

\textbf{Theorem 2.} The set \(\mathcal{H}\) consists of all triples \((I_i, (h(I_i) - I_i), \text{sup}(I_i))\), where \(I_i\) is a frequent key itemset of the given dataset. That is \(\mathcal{H}\) is the closed keys representation of the dataset, with respect to support threshold \(\text{min}\sup\).

The proof of Theorem 2 is immediate by Propositions 4 and 5.

Let us conclude this section with some remarks about the complexities of the closed keys representation and the algorithms. For each triple \((I_i, (h(I_i) - I_i), \text{sup}(I_i))\) in the closed keys representation, the memory space required to store \(I_i\) and \((h(I_i) - I_i)\) can be estimated by the memory space required to store \(h(I_i)\), the closure of \(I\). In each step \(i \geq 2\) of Algorithm \textit{FClosedKeys}, the generation of an \(i\)-itemset candidate \(I_i\), basing on two frequent \((i-1)\)-itemsets \(X, Y, Z\) needs to know the \((i-2)\)-itemsets \(X, Y, Z\) prev, where \(Z\) is a frequent \((i-1)\)-itemset included in \(XY\). These \((i-2)\)-itemsets are associated with the \((i-1)\)-itemsets in step \((i-1)\). Thus in step \(i\), the algorithm needs only the stored information in step \((i-1)\), and not the stored information in step \((i-2)\). The computation of the support of a frequent non-key \(i\)-itemset \(I_i\) is immediate, as well as the field \(I_i, \text{prev}\). Only in the case of frequent key itemset candidates (case b.2 in Algorithm GenCandidate), the computation costs more: checking that all \((i-1)\)-itemsets \(I_{i-1} \subseteq I_i\) are frequent, accessing the dataset to compute the support of \(I_i\), and checking that \(I_i\) is a key itemset. Finally, we observe that in Algorithm \textit{FClosedKeys}, the number of dataset accesses is at most equal to the maximal size of the frequent key itemsets.
5 Related work, Remarks and Conclusions

AClose [9] is an Apriori-like algorithm that computes frequent closed itemsets in two phases. In the first phase, AClose uses a bottom-up search to identify all frequent key itemsets. In the second phase AClose computes the closure of each frequent key itemsets \( I \) as the intersection of all transactions in which \( I \) occurs. Apriori-Close[10] is another Apriori-like algorithm that computes simultaneously frequent and frequent closed itemsets. CHARM [14] is also a bottom-up algorithm that computes frequent closed itemsets. In contrast to AClose, it explores both itemset and transaction identifier set spaces. Moreover, when generating candidates, CHARM avoids enumerating all possible subsets of a closed itemset. CLOSET [11] is a recursive method for computing closed itemsets, that uses a compact representation of transactions, called the FP-tree, where each branch represents the transactions having a same prefix. Dually to Apriori-Close, Pascal [4] computes frequent and frequent key itemsets, with optimization by inferencing supports: Let \( I_k \) be a generated \( k \)-itemset, and \( p_{I_k} \) the minimum of the supports of \((k-1)\)-itemsets \( I_{k-1} \) contained in \( I_k \). If such an \( I_{k-1} \) is not a key itemset, then mark \( I_k \) as non-key, and set \( \text{sup}(I_k) = p_{I_k} \).

Our algorithm is distinct from the above algorithms on the following points:
(i) It computes both frequent key and closed itemsets, and in only one phase.
(ii) It stops as soon as all frequent key itemsets are discovered.
(iii) With respect to inferencing supports, it is distinct from Pascal: When generating candidates in step \( k \geq 2 \), for every candidate \( I_k \), Pascal verifies if all \((k-1)\)-itemsets which are subsets of \( I_k \) are frequent, in our algorithm, this verification only exists for candidates in case (b.2) (see GenCandidate). Moreover, if all \( I_{k-1} \subset I_k \) are frequent, and such an \( I_{k-1} \) is not key, then our algorithm does not compute the minimum. It knows which itemset \( I_{k-1} \subset I_k \) such that \( \text{sup}(I_k) = \text{sup}(I_{k-1}) \).

The disjunction-free sets representation [5] is an approach to concise representations of frequent itemsets, based on the concept of frequent disjunction-free itemset. An itemset \( X \) is called frequent disjunction-free if \( X \) is frequent and there are no items \( A, B \in X \) such that \( X - \{A, B\} \rightarrow A \lor B \) is an exact rule. Otherwise, \( X \) is called disjunctive. Work in [8] applied the notion of disjunction free to generators (key itemsets) to propose a more concise representation, called disjunction-free generators representation. However, computing the disjunction-free generators representation costs more than computing the closed keys representation, in time and in space. Inferencing frequent itemsets basing on the disjunction-free generators representation is a very complex operation and costs more than basing on the closed keys representation. Moreover, the conciseness of the two representations is not comparable.

The above discussion on related work has shown that our approach has many advantages over existing approaches, in terms of algorithms and frequent itemset inferences. In [3] an approach to online mining association rules was proposed, using an adjacency lattice structure to store all frequent itemsets in main memory. This structure allows not only to efficiently mining association rules, but also allows to avoid generating redundant rules. Instead of storing the adjacency lattice structure, we think it will be more efficient if we store the closed keys.
representation in an adequate structure. This is because the closed keys representation is clearly more compact than the adjacency lattice structure, yet the support of any frequent itemset is immediate in the representation, without any computation. The work is currently investigated.

References
1. R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules between Sets of Items in Large Databases", in Proc. of the 1993 ACM SIGMOD Int'l Conf. on Management of Data, May 1993, pp. 207-216.
8. M. Kryszkiewics, "Concise Representation of Frequent Patterns based on Disjunction-free Generators", in Proc. of the 2001 IEEE International Conference on Data Mining (ICDM’01), 29-Nov. - 2-Dec. 2001, San Jose, California, USA, pp. 305-312.
11. J. Pei, J. Han, R. Mao, "CLOSEST: An Efficient Algorithm for Mining Frequent Closed Itemsets", in Proc. Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD), May 2000, p. 21-30.