Abstract: - In this paper we describe an algorithm named MONSA for closed sets mining. It is an exact depth-first search algorithm extracting only frequent closed sets using several pruning techniques to be free from repetitive and empty patterns. The first version of MONSA was created in 1993 and we have developed several versions during last years. Our algorithm does not use such kind of techniques as in ChARM by Zaki and Hsiao. In MONSA there is active only one branch which is under construction. All used techniques are proved. Our purpose is to introduce the approach used in MONSA and the correspondence of its basics and concepts to the approach by Zaki and Hsiao.

We have used MONSA for creating a data mining method called Hypothesis Generator. By the algorithm the intersections (closed sets) and IF…THEN rules on the subsets of source data set simultaneously are found. MONSA does not depend on the initial order of objects. MONSA treats not only binary data, but a larger set of discrete values.

Key-Words: - Data mining, Frequent closed sets, Depth-first search, Monotone systems

1 Introduction

"The task of mining association rules consists of two main steps. The first involves finding the set of all frequent itemsets. The second involves testing and generating all high confidence rules among itemsets." [1] Zaki and Hsiao prove in [1] that "It is not necessary to mine all frequent itemsets in the first step, instead it is sufficient to mine the set of closed frequent itemsets, which is much smaller than the set of all frequent itemsets" and "It is also not necessary to mine the set of all possible rules", because "any rule between itemsets is equivalent to some rule between closed itemsets. Thus many redundant rules can be eliminated."[1] Also it is important to enumerate closed sets directly, without generating them "using Apriori-like bottom-up search methods that examine all subsets of frequent itemsets" or finding them from maximal patterns "since all subsets of the maximal itemsets would again have to be examined"[2].

As we see, for mining association rules we need to find only close sets. We did it. But in our algorithm MONSA we use other denotations and techniques than Zaki and Hsiao [1] [2].

Algorithm MONSA [3] was created for discovering intersections. Appears that the set of intersections it founds is the same as the set of all (frequent) closed sets. Additionally our algorithm enables to find rules (between closed itemsets) at the same time as closed itemsets itself. Nevertheless this is a subset of all possible rules between closed sets. To ensure that these rules hold with at least required confidence it is possible to prune the branches of a search tree according the threshold given by the user.

The main goal of the paper is to introduce our algorithm and the main theoretical conceptions on which it is based.

1.1 The task

MONSA is a depth-first search algorithm used for discovering intersections. The intersection of two (or more) sets is the set of elements, which belong to both (all) sets, simultaneously. Without used pruning techniques MONSA would perform exhaustive search and produce all permutations of all existing value combinations. The value combination is a set of elements, an element is an attribute with certain value. Taking into account the minimal frequency allowed the user only the frequent part of the result is found.

We have to prevent finding 1) repetitious intersections and 2) empty intersections. Finding empty intersections is avoided by the nature of the algorithm, it does not generate combinations, but traverses only through the really existing ones. To prevent “repetitions” we use pruning techniques called “bringing zeros down” and “backward
corresponding frequency table. For example in Table 1 is given a set of two objects described by three attributes and its data set. For example in Table 1 is given a set of two values of attributes appear in frequencies into a frequency table, which shows how frequencies determine a frequent set. We collect the frequencies through the table.

### 1.2 Basic idea of our approach

The fundamental idea of our approach is that frequencies determine a frequent set. We collect the frequencies into a frequency table, which shows how many times different values of attributes appear in data set. For example in Table 1 is given a set of two objects described by three attributes and its corresponding frequency table.

#### Table 1. Example X(2,3)

<table>
<thead>
<tr>
<th>Object \ Attribute</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
</tr>
</thead>
<tbody>
<tr>
<td>O1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>O2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Frequent set = A1.1 AND A3.2

As we see all elements (i.e. attribute with certain value) that have a frequency equal to the number of objects belong to the frequent set. The frequent set with such properties can be found as an intersection through the table.

#### Table 2. Intersection through X(2,3)

<table>
<thead>
<tr>
<th>Object \ Attribute</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
</tr>
</thead>
<tbody>
<tr>
<td>O1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>O2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Intersection \ IntSecX = A1.1 AND A3.2

As we see from Table 1 and Table 2 an intersection is also findable through the frequencies: every value which has frequency equal to the number of objects in the table belongs to the intersection. But we need special techniques to form tables with this property. For every table we find an intersection over the frequency table as we did in Table 1.

We start with definitions in 2. Terms used in closed set mining are given in 2.1 and definitions used explaining MONSA and the relations between terms of two approaches are shown in 2.2. Section 3 describes MONSA, including the algorithm in 3.1, an example of results in 3.2 and the explanation of used pruning techniques in 3.3. Section 4 concludes this paper.

## 2 Definitions

### 2.1 Definitions of closed set mining

Definitions used in closed set mining are presented here as in [1] [2]. Typically the database is arranged as a set of transactions, where each transaction contains a set of items. Let \( I = \{1,2,\ldots,m\} \) be a set of items, and let \( T = \{1,2,\ldots,n\} \) be a set of transaction identifiers or tids.

A set \( X \subseteq I \) is also called an itemset, and a set \( Y \subseteq T \) is called tidset.

\( t(X) \) denotes a tidset that corresponds to an itemset \( X \), i.e. the set of all tids that contain \( X \) as a subset: 
\[
t(X) = \bigcap_{x \in X} \{x\}.
\]

\( i(Y) \) denotes an itemset that corresponds to a tidset \( Y \), i.e. the set of items common to all the tids in \( Y \): 
\[
i(Y) = \bigcap_{y \in Y} i(y).
\]

The support of an itemset \( X \), denoted \( \sigma(X) \), is the number of transactions in which it occurs as a subset.

An itemset is frequent if its support is more than or equal to a user-specified minimum support (\( \text{min\_sup} \) ) value, i.e. if \( \sigma(X) \geq \text{min\_sup} \).

A frequent itemset \( X \) is called closed if there exists no proper superset \( Y \supseteq X \) with \( \sigma(X) = \sigma(Y) \).

Closed sets are found using closure operation. A closure of an itemset \( X \), denoted \( c(X) \), is defined as the smallest closed set that contains \( X \). An itemset \( X \) is closed if and only if \( X = c(X) \).

The closure of an itemset \( X \) is found as: 
\[
c(X) = \{t(t(X))\}.
\]

The support of an itemset \( X \) is also equal to the support of its closure, i.e. \( \sigma(X) = \sigma(c(X)) \).

### 2.2 Definitions used in MONSA

The first version of MONSA was presented in 1993 [3]. Here we present denotations and definitions used for MONSA as in [3]. We also give comments how these notions and concepts relate to the ones used for closed sets (see section 2.1).

1. \( X \) - a set \( X = \{X_i\}, i = 1,2,\ldots,N \), where each object \( X_i \) is a conjunction of M attribute values: 
\[
X_i = \bigwedge_{j=1}^{M} h_{ij}.
\]

\( X \) is a set of \( N \) objects (records) that are described by \( M \) attributes. Set \( X \) has not to be a binary dataset, every attribute \( j \) can acquire integer values in the interval \( h_{ij} = 0,1,2,\ldots,K_j-1 \). \( X \) can be a transaction database also.
(2) $H$ - a value combination (VC) of certain attributes $H = \& h_q, D = \{ j_e \}, e = 1, \ldots, E_H$ (number of elements $h_q$ in $H$), $1 \leq E_H \leq M$, $1 \leq j_e \leq M$, $j_n j_i \in D, j_r \neq j_i, f \# t, H \subseteq X_i$.

A value combination can contain 1 to $M$ attributes (with certain values), each only once (i.e. only with one value); it is a subset of some object or is a whole object.

(3) Each value combination $H$ defines on the set $X$ a subset of objects $X_H = \{ X_p \}$, $p = 1, 2, \ldots, N_H$, $1 \leq N_H \leq N$.

$\{ X_p \}$ are all objects $X_i \in X$ that contain $H$: $X_H = \{ X_i \in X | X_i \supseteq H \}$.

The subset of objects defined by the value combination is similar to the tidset that corresponds to some itemset ($t(X)$).

(4) Each value combination $H$ defines on set $X$ a subset of elements $X_H \subseteq X$: $X_H = \{ X_{ij} \} \in X, i = 1, 2, \ldots, N, j = 1, 2, \ldots, M$.

An attribute with certain value is called element. ‘Element’ corresponds to ‘item’. The difference is that each attribute produces as many elements as many different values it has. Definition (4) says that ‘value combination’ is the same as ‘itemset’ (with extension that values have not to be binary).

(5) Intersection over a set $Y = \{ Y_t \}$, $t = 1, 2, \ldots, T$, $Y_t = \& h_q$ is a set of such elements $h_q$ which belong simultaneously to all $Y_t$: $\bigcap_{t=1}^{T} Y_t = \& h_q = H$.

In $Y$ for $H$ there exists always a corresponding subset of objects $Y_H = \{ Y_p \}$, $p = 1, \ldots, N_H$, $N_H \leq N$.

If $N = 1$, then intersection over $Y$ is an object itself.

If there exist no objects $Y_t \in Y$ for which $H \subset Y_t$, then $Y_H = \emptyset$.

Definition (5) says that intersection over a set of objects is a set of common elements, this is the same as itemset that corresponds to some tidset ($t(X)$).

(6) Elementary conjunction (EC) on $X_H$ is such an intersection over the set $X_H$, where $\cap X_H = A(\supseteq H)$, $X_A = X_H$, $N_H \leq N$.

In the case of $A \supseteq H$, $X_A = X_H$, $A$ is an EC.

We have a set of elements $H$ and its corresponding set of objects $X_H$ (i.e. $t(H)$) and find an intersection over it: $\cap X_H$ (i.e. $t(t(H))$). The operation $t(t(H))$ means finding the closure of $H$. Therefore the resultant set (of elements) $A$ called elementary conjunction is a closed set. From the viewpoint of the algorithm it is essential to find a technique that guarantees the finding of such subsets $X_H$ only for which $\cap X_H = A(\supseteq H)$ (i.e. finding of closed sets only).

(7) Maximal EC on $X_H$ is such an intersection over $X_H$ in case of which for a VC $H = \& h_q$ is a valid relation

$\cap X_H = A(= h_q) \supseteq H$, $1 \leq q \leq e \leq M, X_A = X_H$.

By definition, VC $H$ is EC if $\cap X_H = H$.

$H$ is a maximal EC if it is EC and contains at least one VC $Ht \subset H$ such, that $|X_{Ht}| = |X_H|$ on $X$. That means that the set of objects $X_H \subseteq X$ is defined unique.

Definition (7) says that maximal EC is EC that has at least one (non-closed) subset with the same frequency (support) i.e. we can remove at least one element without changing in frequency. Our maximal elementary conjunction here is not the same as maximal (closed) set'.

Additionally, our ‘(absolute) frequency’ is the same as ‘support’.

For rules we use also ‘relative frequency’ which corresponds to ‘confidence’.

3 MONSA

MONSA is a depth-first search algorithm. Without pruning techniques it would traverse the complete search tree and find all permutations of all existing value combinations. Empty (i.e. non-existing) combinations are avoided by nature of algorithm, they are not generated (and checked for existence) at all.

From the initial data set MONSA finds a result as a set of intersections (closed sets) and/or a set of trees (forest), they are listed in the order they are found, that order does not depend on the initial order of objects (records). The frequencies of nodes (intersections) decrease strictly along branches of a tree(s). The decrease makes allowable to prune the branches according to the minimal allowed frequency (support). This is similar to the other tree-based algorithms, including ChARM [2]. The fact that the decrease is strict gives a high potential to the intersection (combination from root to the certain node) to be a closed set. At any level the descendants of a common parent-node are found in a weakly decreasing order of their frequencies, the roots also are found in a weakly descending order. The order of nodes with equal frequency depends on the searching principle (usually by columns or by rows of frequency table).

\footnote{\textsuperscript{1}“A frequent itemset $X$ is called maximal if it is not a subset of any other frequent itemset.” [2]}
MONSA uses frequency tables (introduced in section 1.2). For every extract of objects its corresponding frequency table is formed.

3.1 Description of the algorithm
MONSA finds intersections in given set X(N,M), where N is the number of objects (for example transactions), M is the number of attributes and each attribute j has an integer value h_j=0,1,2,...,K-1. A pair of attribute and its certain value is called element.

By essence MONSA is a recursive algorithm. Here its backtracking version is presented.

In this algorithm the following notation is used:
- t: the number of the step (or level) of the recursion
- FT_t: frequency table for a set X_t
- IntSec_t: vector of elements over set X_t (intersection)
- Init: activity for initial evaluation

Algorithm MONSA

Init  
\( t \leftarrow 0, \text{IntSec}_0 \leftarrow \{\} \)

Find a table of frequencies FT_0 for all attributes in X_0
Do while there exists FT_s≠Ø in \( \{FT_s\} \), s≤t
  - For each element h_s∈FT_s with frequency V=max FT_s(h_s)#0 do
    - If pruning is needed (h_s has to be pruned) then goto back
    - Separate submatrix X_{t+1}⊂X_t such that \( X_{t+1} = \{X_{ij} \in X_t | X.f = h_s\} \)
    - Find a table of frequencies on X_{t+1} FT_{t+1}
    - ZeroesDown(t+1)
    - BackwardComparison(t+1)
    - Output of IntSec_{t+1}
    - If new intersection is unique then add elements j with FT_{t+1}(j)=V into vector IntSec_{t+1}
    - If there exist attributes to analyse then t=t+1
  Next
Back: t=t-1
IntSec_{t+1}←IntSec_t
Enddo
All intersections are found
End: end of algorithm

Elimination (pruning) activities:

1) ZeroesDown(t+1)
   For each element h_s∈FT_t do
   If FT_t(h_s)=0 then FT_{t+1}(h_s)←0
   Next

2) BackwardComparison(t+1)
   For each element h_s∈FT_{t+1} with frequency #0 do
   If FT_{t+1}(h_s)=FT_t(h_s) then FT_t(h_s)←0
   Next

3) CheckUniqueness(t+1)
   If there exists on X_{t+1} h_u, 1≤u≤M such, that \[ h_u \in \text{IntSec}_{t+1} \text{ AND } FT_{t+1}(h_u)=0 \text{ AND } \]
   frequency of h_u in X_{t+1}=V \] then
   Intersection is not unique
   Else
   Intersection is unique
   Endif

It has been proved that if a finite discrete data matrix X(N,M) is given, where N=K^M, then the complexity of algorithm MONSA to find all (K+1)^M elementary conjunctions (intersections) as existing value combinations is \( O(N^2) \) operations [3].

By our estimation in practice the upper bound of the number of value combinations (with minimal frequency allowed = 1) is

\[
L_{\text{UP}} \approx N(1+1/K)^M, \quad (1)
\]

but usually it is less.

The precise formula for the number of intersections is as follows:

\[
L = \sum_{f=1}^{F} \left( \sum_{p=1}^{M*K-u} N_p \right), \quad (2)
\]

where F is the number of formatted frequency tables on set X, u is the number of empty cells in the frequency table FT_t, N_p is the absolute number in a cell of the frequency table (frequency of an attribute value).

The most important advantages of the algorithm are:

- The frequency is known at the moment the new node is found
- Ability to find nodes consisting of more than one element, this reduces the number of nodes and the size of the tree
3.2 Example of results
Let us have a data set given in Table 3.

<table>
<thead>
<tr>
<th>Object</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>A4</th>
</tr>
</thead>
<tbody>
<tr>
<td>O1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>O2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>O3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>O4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

From these data MONSA finds 5 intersections (with minimal frequency allowed =2):

1. \(0**\) \(A1.1&A2.0=3\)
2. \(101*\) \(A1.1&A2.0&A3.1=2\)
3. \(10*1\) \(A1.1&A2.0&A4.1=2\)
4. \(**1*\) \(A3.1=3\)
5. \(**10\) \(A3.1&A4.0=2\)

They all are different, and they are not empty. We can affirm that there is no special effort to check the intersection’s uniqueness during the extraction process. Repetitions are prevented just by using branches’ pruning techniques.

Instead of list of intersections the same result can be listed as a set of trees (immediately during the process of finding intersections):

\[(3) \quad 0.667 (2)\]
\[A1.1&A2.0=>A3.1\]
\[0.667 (2)\]
\[=>A4.1\]
\[(3) \quad 0.667 (2)\]
\[A3.1=>A4.0\]

Trees are represented from left to right. This fragment consists of two trees, it has two root nodes (on the left). Symbols ‘\=>’ separate nodes of a tree.

Usually a node contains a pair of certain attribute and a certain value of that attribute. Attribute is shown before period and value is after period. For example, the root node of the second tree contains attribute A3 and its value 1. A node can consist of more than one attribute-value pairs like the root node of the first tree, then ‘&’ is used to connect them.

The numbers above node show frequencies.

In the parentheses node’s absolute frequency is shown. Absolute frequency of node \(t\) shows how many objects have a certain attribute with a certain value among objects having properties (i.e. certain attributes with certain values) of all previous levels \(t-1,\ldots,1\). For example, the second tree shows that there are two objects having A4.0 (i.e. attribute A4 with value 0) among objects with A3.1.

Before parentheses node’s relative (to previous level) frequency is shown. Relative frequency is a ratio \(A/B\), where \(A\) is the absolute frequency of node \(t\) and \(B\) is the absolute frequency of node \(t-1\). For the first level the relative frequency is not calculated.

3.3 Pruning techniques used in MONSA
In order to find intersections MONSA uses frequency tables. A frequency table contains the counts of occurrences of all existing values for each attribute. Each attribute can have a different number of different discrete values. In our previous examples (in sections 1.2 and 3.2) we have two different values for each attribute, but this is not a limitation by the algorithm.

Zero in the initial frequency table means that the corresponding element does not exist. During the work the frequency tables are used to collect information about pruning also. Elements that are exhaustively analyzed are zerofilled in frequency tables. Such prohibited (eliminated) elements are not included into the intersections any more, only the elements with frequency over zero (or some higher threshold) are considered.

To avoid finding “repetitions” i.e. permutations of already found intersections two pruning techniques have been used in algorithm MONSA:

“bringing zeroes down” – activity that prohibits arbitrary output repetition of already separated intersection at the next (deeper) level(s);

“backward comparison” – activity that does not allow the output of the separated intersection at the same (current) level and also at previous (higher) levels (after backtracking).

Impact of these techniques is proved in [3].

Appears that these two activities do not prevent repetitious finding (and output) of some subsets of already found intersections.

The subset of already found intersection is redundant only if both have equal frequencies (i.e. this (sub)set is non-closed). If subset’s frequency is higher (than its superset’s) then it covers more objects and is not redundant. Subset’s frequency cannot be lesser.

Finding a superset of already found set with the same frequency is impossible, because at any level MONSA finds all co-existing (i.e. contained in the same objects) elements with equal frequencies as one intersection (i.e. maximal EC).

---

2 Theorems 5.3 and 5.4 accordingly
Next we give an example of redundant subsets under consideration. For that we have used MONSA without “uniqueness check” (of a new intersection).

Having the initial data set of nine objects described by three attributes (see Table 4) MONSA (without uniqueness check) finds fifteen intersections. Both representation forms – trees and intersections – are given in Figure 1.

Table 4. Example X(9,3)

<table>
<thead>
<tr>
<th>Object \ Attribute</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
</tr>
</thead>
<tbody>
<tr>
<td>O1</td>
<td>4</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>O2</td>
<td>5</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>O3</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>O4</td>
<td>5</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>O5</td>
<td>3</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>O6</td>
<td>3</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>O7</td>
<td>3</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>O8</td>
<td>3</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>O9</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 4. Example X(9,3)

(a) Result as a set of trees:

<table>
<thead>
<tr>
<th>(4)</th>
<th>0.500 (2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1.5 =&gt; A2.4 &amp; A3.3</td>
<td></td>
</tr>
<tr>
<td>0.250 (1)</td>
<td></td>
</tr>
<tr>
<td>=&gt; A2.0 &amp; A3.7</td>
<td></td>
</tr>
<tr>
<td>0.250 (1)</td>
<td></td>
</tr>
<tr>
<td>=&gt; A2.3 &amp; A3.4</td>
<td></td>
</tr>
</tbody>
</table>

(b) Result as a set of intersections:

| (11) | A1.5=4 |
| (12) | A1.5 & A2.4 & A3.3 =2 |
| (13) | A1.5 & A2.0 & A3.7 =1 |
| (14) | A1.5 & A2.3 & A3.4 =1 |

Figure 1. Result found by MONSA (without uniqueness check)

Here we have six trees and six roots accordingly. Intersections 11, 15, 18, 111, 113 and 115 correspond to the roots.

Intersection II1 (A2.0=3) is not redundant although A2.0 is contained in the intersections I3, I7 and I9, because these three intersections have lesser frequencies and none of them cover all objects covered by I11.

Intersections I9, I12 and I14 are redundant:

- I9 (A1.4 & A2.0) is a subset of I7 (A3.8 & A1.4 & A2.0) with the same frequency =2 (both cover objects O1 and O8);
- I12 (A2.0 & A3.7) is a subset of I3 (A1.5 & A2.0 & A3.7), both frequencies are 1 (they cover object O4) and
- I14 (A2.1 & A3.7) is a part of I10 (A1.4 & A2.1 & A3.7), frequencies equal 1 (they cover O7).

There have to be 12 intersections instead of 15.

Starting from the root of a tree the frequencies of intersections (nodes) always (strictly) decrease along any branch of the tree (due to finding maximal EC at every node). As no branch has two intersections with the same frequency, the redundant subsets do not occur in the same branch, they can appear in different branches of a tree or in different trees.

Among siblings (i.e. direct descendants of a common node) any element can appear in only one intersection. Consequently, redundant subsets (under consideration) do not occur among siblings. This is also true for the root-level (which formally consists of descendants of the initial empty set), therefore these redundant subsets never occur at root-level.

Intersection (closed set) and its redundant subintersection (with the same frequency) do not have to appear at the same level of a tree (as in all three cases of our example).

Element(s) that appear in the root node are eliminated from further analysis by zero-filling the corresponding cell(s) in the frequency table. Elements that are fully analyzed (exhausted) at deeper levels are prohibited by “backward comparison”. All these zeroes are “brought down” to all succeeding levels and therefore prohibited elements never occur in redundant intersections. So (due to the elimination techniques used) no permutation of whole (already found) intersection (closed set) is not found. Elements that are partially analyzed (at the non-root levels) are not eliminated. All this is true for any subtree also.

Although prohibited elements are eliminated from the frequency tables they still appear in the subsets of objects extracted by non-prohibited elements. Remind that some set (of elements) and its subset with the same frequency define the same set of objects. If some prohibited (and eliminated) element appears in all objects (of subset) it means that this subset has been analyzed already (this element itself can not be contained in the value combination (potential intersection) on which basis that subset of
objects was extracted). All intersections containing a prohibited element are already found and such subsets need no more analysis. This situation indicates a repetitive extraction of certain subset of objects.

To exclude redundant subsets (sub-intersections) we have to detect a situation when some prohibited element occurs in all objects and stop analyzing such branch.

In our example (see Table 4 and Figure 1) intersection I7 (A3.8&A1.4&A2.0) has one more element than redundant set I9 (A1.4&A2.0), namely A3.8. The set of objects extracted by I9 is the same as by I7 (objects O1 and O8). Consequently, actual frequency of A3.8 is as much as number of objects in that set (=2). As A3.8 was prohibited after exhaustive analyze, the frequency table contains zero in the corresponding cell. Detecting such situation we can say that A1.4&A2.0 (I9) is a redundant set.

Such “uniqueness check” is used by MONSA. It is not necessary to look through the already found intersections to ensure the new one is non-redundant.

Correctness of ‘uniqueness check’ explained here is proved in [4].

It is interesting that those redundant subsets seem to be the same ones for what ChARM [1] [2] needs “subsumption checking”. In order to ensure that a candidate set is really closed ChARM looks through the (certain) already found closed sets, only those that have 1) the same “tidsum” and 2) the same support (frequency) as the candidate set. (Tidsums of different closed sets with equal frequency tend to be different). For the complete description see [1] [2].

As shown already we perform the uniqueness check of a new intersection (potential closed set) otherwise, without looking through the already found (closed) sets.

4 Conclusion
In this paper we have described algorithm MONSA for finding intersections that is the same as finding closed sets. The paper introduces the approach used in MONSA and the correspondence of its basics and concepts to the approach by Zaki and Hsiao (algorithm ChARM).

MONSA uses effective pruning techniques to prevent repetitious finding of already found intersections (closed sets) and uniqueness (non-redundancy) check of a new potential intersection without looking through the already found intersections. While finding all intersections (closed sets) MONSA can find also rules between these intersections. At the same time the algorithm works not only on the binary data.

Used concepts are compared to the ones used by Zaki and Hsiao and it is shown that substantially they are the same.

MONSA is a basic algorithm we have used not only in data mining [5] [6] [7], but also in solving of graph theoretical problems as extracting of all maximal cliques [4] and decision trees constructing [8]. We have some approaches how to use the algorithm presented here in machine learning for the future works.

Acknowledgement
This work was supported in part by the Estonian Information Technology Foundation under Grant 07-03-00-22.

References: