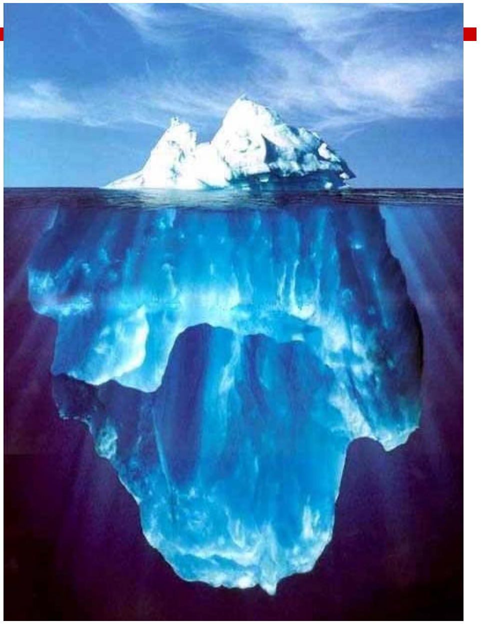
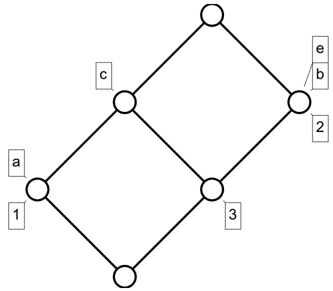
Formal Concept Analysis

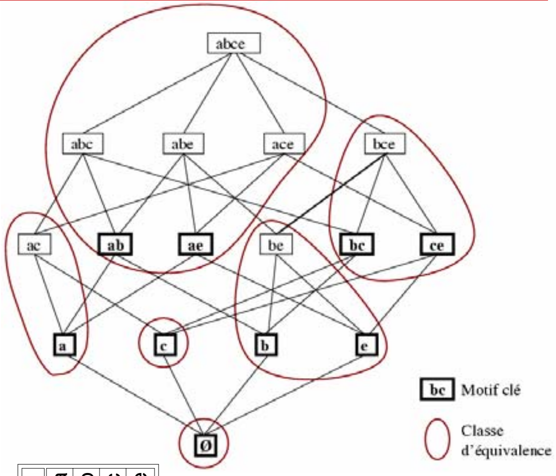
2 Closure Systems and Implications

4 Closure Systems



Concept intents as closed sets





	a	b	c	e
1	×		×	
2		×		×
3		×	×	×

Next-Closure

was developed by B. Ganter (1984).

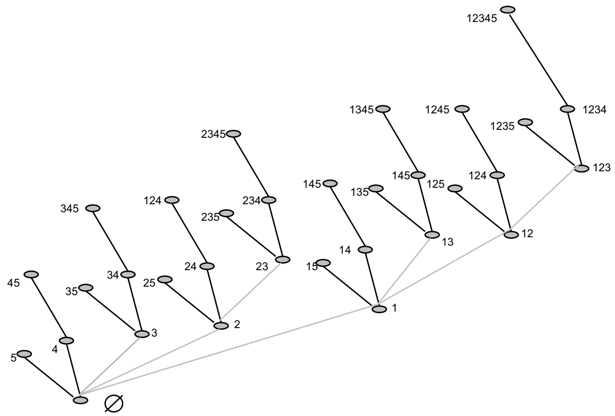
It can be used

- to determine the concept lattice or
- to determine the concept lattice together with the stem basis or
- for interactive knowledge acquisition.

It determines the concept intents in **lectical order.**

Let $M = \{1, ..., n\}$. $A \subseteq M$ is **lectically smaller** than $B \subseteq M$, if $B \neq A$ if the smallest element where A and B differ belongs to B :

$A < B \;:\Leftrightarrow\; \exists\, i \in B\backslash A\colon A \cap \{1, 2, ..., i\text{-}1\} = B \cap \{1, 2, ..., i\text{-}1\}$



We need the following:

$A <_i B \;:\Leftrightarrow\; i \in B \setminus A \;\wedge\; A \cap \{1, 2, ..., i\text{-}1\} = B \cap \{1, 2, ..., i\text{-}1\}$

$A \bullet i \;:= (A \cap \{1, 2, ..., i\text{-}1\}) \cup \{i\}$

Theorem: The smallest concept intent, which according to the lectical order is larger as a given set $A \subset M$, is

$$A \oplus i \;:= (A \bullet i)'',$$

where i is the largest element of M with $A <_i A \oplus i$.

Algorithm **Next-Closure** for determining all concept intents:

1) The lectically smallest concept intent is $\varnothing''$.

2) Is A a concept intent, then we find the lectically next intent, by checking all attributes $i \in M \setminus A$, starting with the largest, und then in decreasing order, until $A <_i (A \oplus i)''$ holds. Then $A \oplus i$ is the lectically next concept intent.

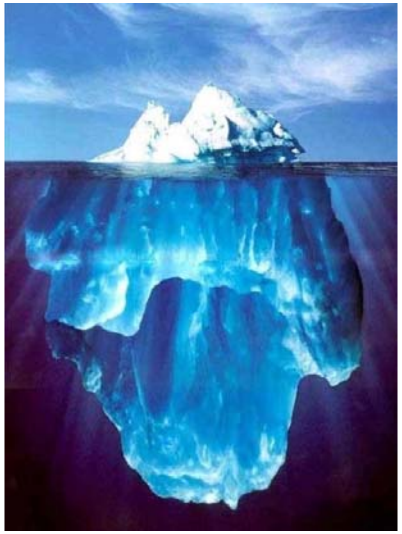
3) If $A \oplus i = M$, then stop, else $A \leftarrow A \oplus i$ and goto 2).

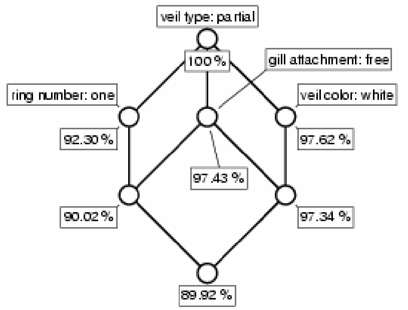
	Handy (1)	Telefon (2)	Fax (3)	Fax w. n. paper (4)
Sinus 44		X		
Nokia 6110	X	X		
T-Fax 301			X	X
T-Fax 360 PC			X	

Example: on blackboard

A	i	$A \bullet i$	$A \oplus i := (A \bullet i)''$	$A <_i A \oplus i$?	new concept intent

Iceberg Concept Lattices

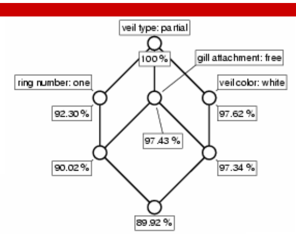




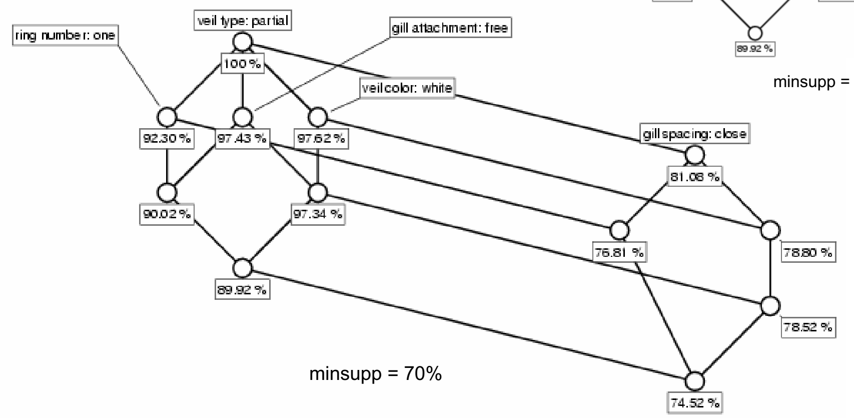
minsupp = 85%

For minsupp = 85% the seven most general of the 32.086 concepts of the Mushrooms database http:\\kdd.ics.uci.edu are shown.

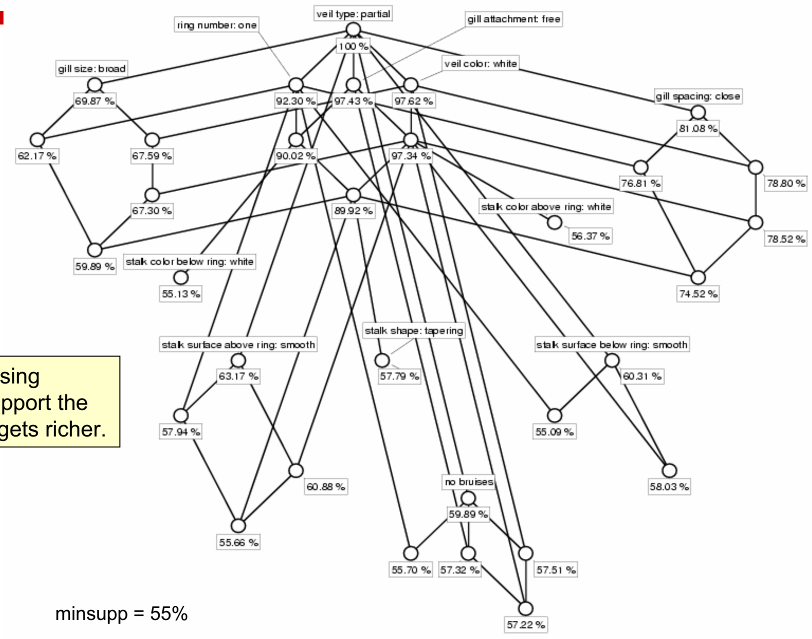
Iceberg Concept Lattices



minsupp = 85%

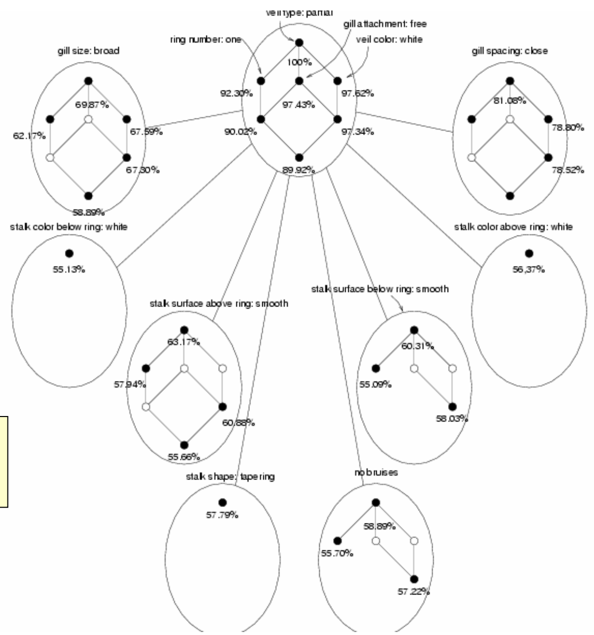


minsupp = 70%



With decreasing minimum support the information gets richer.

minsupp = 55%



The visualization as a nested line diagram indicates implications.

The support of a set $X \subseteq M$ of attributes is given by

$$\mathrm{supp}(X) = \frac{|X'|}{|G|}$$

• Def.: The **iceberg concept lattice** of a formal context (G,M,I) for a given minimal support minsupp is the set

$$\{ (A,B) \in \underline{\mathbf{B}}(G,M,I) \mid \mathrm{supp}(B) \geq \text{minsupp} \}$$

• It can be computed with **TITANIC.** [Stumme et al 2001]

TITANIC

computes the closure system of all (frequent) concept intents using the *support function*:

Def.: The **support** of an attribute set (itemset) $X \subseteq M$ is given by

$$\text{supp}(X) = \frac{|X'|}{|G|}$$

Only concepts with a support above a threshold minsupp $\in [0,1]$.

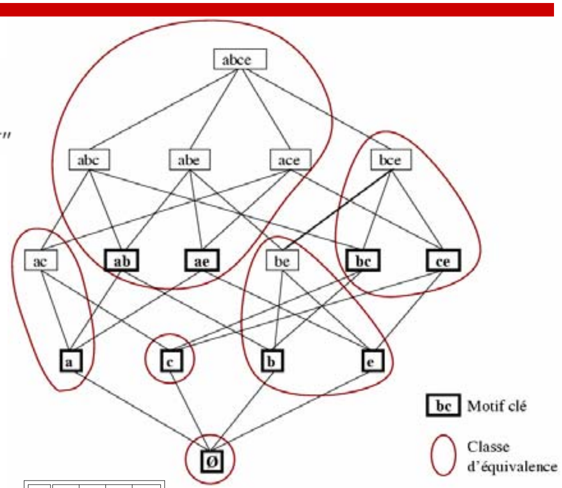
TITANIC makes use of some simple facts about the support function:

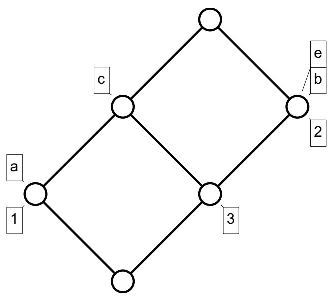
Lemma 4. *Let $X, Y \subseteq M$.*

1. $X \subseteq Y \Longrightarrow \text{supp}(X) \geq \text{supp}(Y)$
2. $X'' = Y'' \Longrightarrow \text{supp}(X) = \text{supp}(Y)$
3. $X \subseteq Y \wedge \text{supp}(X) = \text{supp}(Y) \Longrightarrow X'' = Y''$

Lemma 4. *Let $X, Y \subseteq M$.*

1. $X \subseteq Y \Longrightarrow \text{supp}(X) \geq \text{supp}(Y)$
2. $X'' = Y'' \Longrightarrow \text{supp}(X) = \text{supp}(Y)$
3. $X \subseteq Y \wedge \text{supp}(X) = \text{supp}(Y) \Longrightarrow X'' = Y''$





	a	b	c	e
1	×		×	
2		×		×
3		×	×	×

TITANIC

tries to optimize the following three questions:

1. How can the closure of an itemset be determined based on supports only?
2. How can the closure system be computed with determining as few closures as possible?
3. How can as many supports as possible be derived from already known supports?

TITANIC

1. How can the closure of an itemset be determined based on supports only?

$X'' = X \cup \{ x \in M \setminus X \mid supp(X) = supp(X \cup \{ x \}) \}$

Example: { b,c }“ = { b, c, e }, since

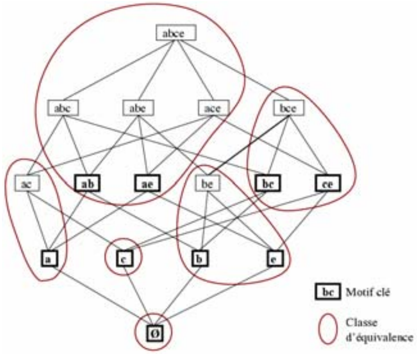
supp({ b, c }) = 1/3

and

supp({ a, b, c }) = 0/3

supp({ b, c, e }) = 1/3,

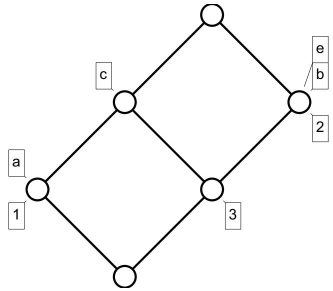
	a	b	c	e
1	×		×	
2		×		×
3		×	×	×

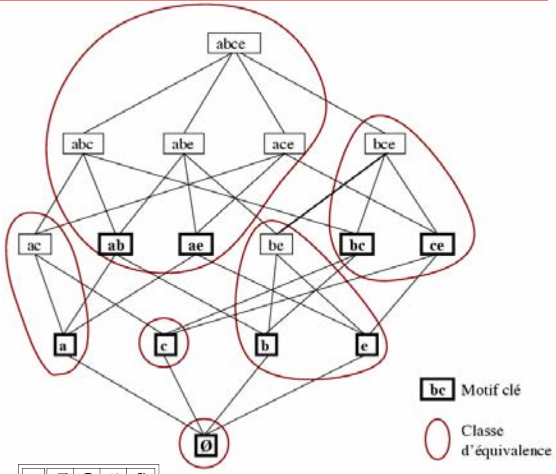


TITANIC

2. How can the closure system be computed with determining as few closures as possible?

- We determine only the closures of the minimal generators.





	a	b	c	e
1	×		×	
2		×		×
3		×	×	×

TITANIC

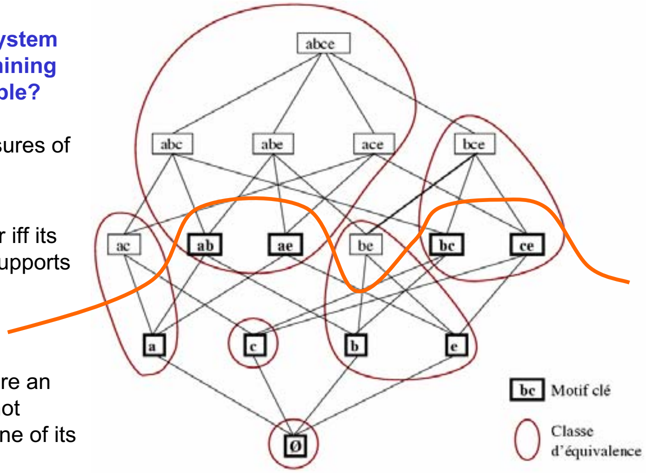
2. How can the closure system be computed with determining as few closures as possible?

We determine only the closures of the minimal generators.

- A set is minimal generator iff its support is different of the supports of all its lower covers.

- The minimal generators are an order ideal (i.e., if a set is not minimal generator, then none of its supersets is either.)

→ Apriori like approach



In the example, TITANIC needs two runs (and Apriori four).

TITANIC

1. How can the closure of an itemset be determined based on supports only?

X“ = X ∪ x∈ M \ X | supp(X) = supp(X ∪ x)

2. How can the closure system be computed with determining as few closures as possible?

Approach à la Apriori

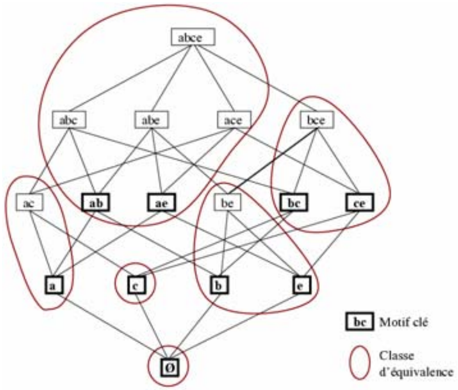
3. How can as many supports as possible be derived from already known supports?

3. How can as many supports as possible be derived from already known supports?

	a	b	c	e
1	×		×	
2		×		×
3		×	×	×

Theorem: If X is no minimal generator, then

supp(X) = min { supp(K) | K is minimal generator, $K \subseteq X$ } .



Example: supp({ a, b, c }) = min { 0/3, 1/3, 1/3, 2/3, 2/3 } = 0, since the set is no minimal generator, and since

supp({ a, b }) = 0/3, supp({ b, c }) = 1/3
supp({ a }) = 1/3, supp({ b }) = 2/3
supp({ c }) = 2/3

Remark: It is sufficient to check the largest generators K with $K \subseteq X$, i.e. here { a, b } and { b, c} .

TITANIC

1. How can the closure of an itemset be determined based on supports only?

X" = X ∪ { x∈ M \ X | supp(X) = supp(X ∪ x) }

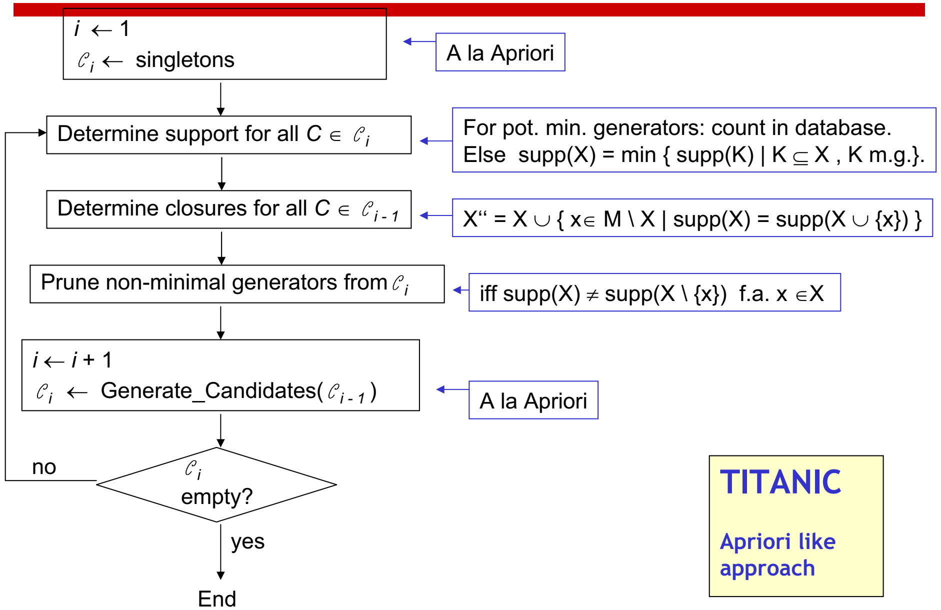
2. How can the closure system be computed with determining as few closures as possible?

Approach à la Apriori

3. How can as many supports as possible be derived from already known supports?

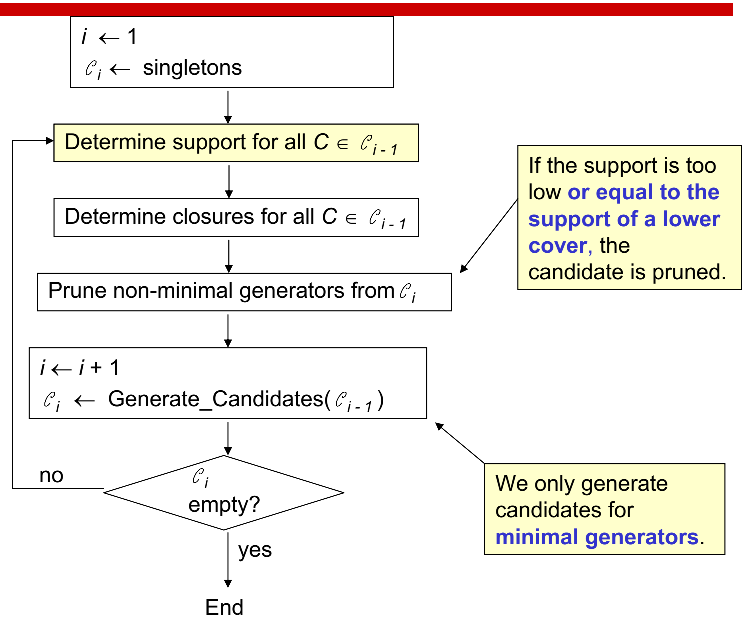
If X is no minimal generator, then

supp(X) = min { supp(K) | K is minimal generator, $K \subseteq X$ } .



TITANIC

compared with Apriori



TITANIC

Algorithm 1 TITANIC

1) WEIGH($\{\emptyset\}$);
2) $\mathcal{K}_0 \leftarrow \{\emptyset\}$;
3) $k \leftarrow 1$;
4) **forall** $m \in M$ **do** $\{m\}.p\_s \leftarrow \emptyset.s$;
5) $\mathcal{C} \leftarrow \{\{m\} \mid m \in M\}$;
6) **loop begin**
7) WEIGH($\mathcal{C}$);
8) **forall** $X \in \mathcal{K}_{k-1}$ **do** X.closure $\leftarrow$ CLOSURE(X);
9) $\mathcal{K}_k \leftarrow \{X \in \mathcal{C} \mid X.s \neq X.p\_s\}$;
10) **if** $\mathcal{K}_k = \emptyset$ **then exit loop** ;
11) $k++$;
12) $\mathcal{C} \leftarrow$ TITANIC-GEN($\mathcal{K}_{k-1}$);
13) **end loop** ;
14) **return** $\bigcup_{i=0}^{k-1}\{X.\text{closure} \mid X \in \mathcal{K}_i\}$.

k is the counter which indicates the current iteration. In the kth iteration, all key k-sets are determined.
$\mathcal{K}_k$ contains after the kth iteration all key k-sets K together with their weight $K.s$ and their closure K.closure.
$\mathcal{C}$ stores the candidate k-sets C together with a counter $C.p\_s$ which stores the minimum of the weights of all $(k-1)$-subsets of C. The counter is used in step 9 to prune all non-key sets.

TITANIC

Algorithm 2 TITANIC-GEN

Input: $\mathcal{K}_{k-1}$, the set of key $(k-1)$-sets K with their weight $K.s$.

Output: $\mathcal{C}$, the set of candidate k-sets C
with the values $C.p\_s := \min\{s(C \setminus \{m\} \mid m \in C\}$.

The variables p_s assigned to the sets $\{m_1, \ldots, m_k\}$ which are generated in step 1 are initialized by $\{m_1, \ldots, m_k\}.p\_s \leftarrow s_{\max}$.

1) $\mathcal{C} \leftarrow \{\{m_1 < m_2 < \ldots < m_k\} \mid \{m_1, \ldots, m_{k-2}, m_{k-1}\}, \{m_1, \ldots, m_{k-2}, m_k\} \in \mathcal{K}_{k-1}\}$;
2) **forall** $X \in \mathcal{C}$ **do begin**
3) **forall** $(k-1)$-subsets S of X **do begin**
4) **if** $S \notin \mathcal{K}_{k-1}$ **then begin** $\mathcal{C} \leftarrow \mathcal{C} \setminus \{X\}$; **exit forall** ; **end**;
5) $X.p\_s \leftarrow \min(X.p\_s, S.s)$;
6) **end**;
7) **end**;
8) **return** $\mathcal{C}$.

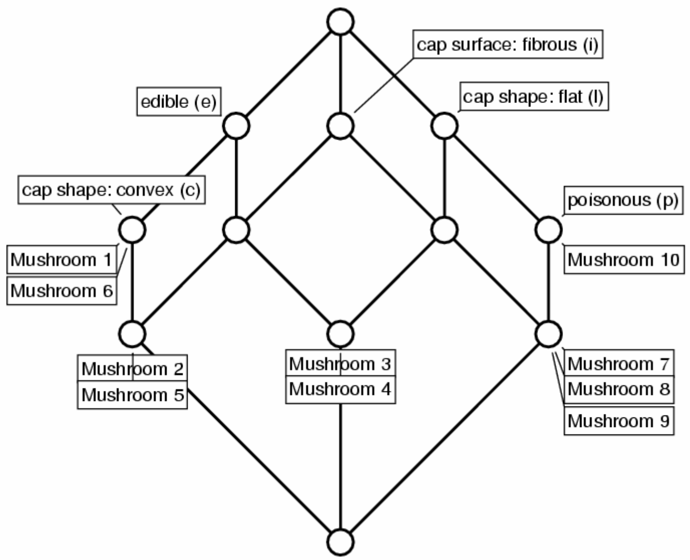
TITANIC

Algorithm 3 CLOSURE(X) for $X \in \mathcal{K}_{k-1}$

1) $Y \leftarrow X$;
2) **forall** $m \in X$ **do** $Y \leftarrow Y \cup (X \setminus \{m\})$.closure;
3) **forall** $m \in M \setminus Y$ **do begin**
4) **if** $X \cup \{m\} \in \mathcal{C}$ **then** $s \leftarrow (X \cup \{m\}).s$
5) **else** $s \leftarrow \min\{K.s \mid K \in \mathcal{K}, K \subseteq X \cup \{m\}\}$;
6) **if** $s = X.s$ **then** $Y \leftarrow Y \cup \{m\}$
7) **end**;
8) **return** Y.

Example of TITANIC

	edible (e)	poisonous (p)	cap shape: convex (c)	cap shape: flat (l)	cap surface: fibrous (i)
Mushroom 1	X		X		
Mushroom 2	X		X		X
Mushroom 3	X			X	X
Mushroom 4	X			X	X
Mushroom 5	X		X		X
Mushroom 6	X		X		
Mushroom 7		X		X	X
Mushroom 8		X		X	X
Mushroom 9		X		X	X
Mushroom 10		X		X	



$k = 0$:

step 1		step 2
X	$X.s$	$X \in \mathcal{K}_k$?
$\emptyset$	1	yes

$k = 1$:

steps 4+5	step 7	step 9
X	$X.p\_s$	$X.s$	$X \in \mathcal{K}_k$?
$\{e\}$	1	6/10	yes
$\{p\}$	1	4/10	yes
$\{c\}$	1	4/10	yes
$\{l\}$	1	6/10	yes
$\{i\}$	1	7/10	yes

Step 8 returns: $\emptyset$.closure $\leftarrow \emptyset$

	edible (e)	poisonous (p)	cap shape: convex (c)	cap shape: flat (l)	cap surface: fibrous (i)
Mushroom 1	×		×		
Mushroom 2	×		×		×
Mushroom 3	×			×	×
Mushroom 4	×			×	×
Mushroom 5	×		×		×
Mushroom 6	×		×		
Mushroom 7		×		×	×
Mushroom 8		×		×	×
Mushroom 9		×		×	×
Mushroom 10		×		×	

Then the algorithm repeats the loop for $k = 2, 3$, and 4:

$k = 2$:

step 12	step 7	step 9	
X	$X.p\_s$	$X.s$	$X \in \mathcal{K}_k$?
$\{e,p\}$	4/10	0	yes
$\{e,c\}$	4/10	4/10	no
$\{e,l\}$	6/10	2/10	yes
$\{e,i\}$	6/10	4/10	yes
$\{p,c\}$	4/10	0	yes
$\{p,l\}$	4/10	4/10	no
$\{p,i\}$	4/10	3/10	yes
$\{c,l\}$	4/10	0	yes
$\{c,i\}$	4/10	2/10	yes
$\{l,i\}$	6/10	5/10	yes

Step 8 returns: $\{e\}$.closure $\leftarrow \{e\}$
$\{p\}$.closure $\leftarrow \{p,l\}$
$\{c\}$.closure $\leftarrow \{c,e\}$
$\{l\}$.closure $\leftarrow \{l\}$
$\{i\}$.closure $\leftarrow \{i\}$

	edible (e)	poisonous (p)	cap shape: convex (c)	cap shape: flat (l)	cap surface: fibrous (i)
Mushroom 1	×		×		
Mushroom 2	×		×		×
Mushroom 3	×			×	×
Mushroom 4	×			×	×
Mushroom 5	×		×		×
Mushroom 6	×		×		
Mushroom 7		×		×	×
Mushroom 8		×		×	×
Mushroom 9		×		×	×
Mushroom 10		×		×	

$k = 3$:

step 12	step 7	step 9	
X	$X.p\_s$	$X.s$	$X \in \mathcal{K}_k$?
$\{e,l,i\}$	2/10	2/10	no
$\{p,c,i\}$	4/10	0	yes
$\{c,l,i\}$	4/10	0	yes

Step 8 returns: $\{e,p\}$.closure $\leftarrow \{e,p,c,l,i\}$
$\{e,l\}$.closure $\leftarrow \{e,l,i\}$
$\{e,i\}$.closure $\leftarrow \{e,i\}$
$\{p,c\}$.closure $\leftarrow \{e,p,c,l,i\}$
$\{p,i\}$.closure $\leftarrow \{p,l,i\}$
$\{c,l\}$.closure $\leftarrow \{e,p,c,l,i\}$
$\{c,i\}$.closure $\leftarrow \{e,c,i\}$
$\{l,i\}$.closure $\leftarrow \{l,i\}$

$k = 4$:

Step 12 returns the empty set. Hence there is nothing to weigh in step 7. Step 9 sets $\mathcal{K}_4$ equal to the empty set; and in step 10, the loop is exited.

Step 8 returns: $\{p,c,i\}$.closure $\leftarrow \{e,p,c,l,i\}$
$\{c,l,i\}$.closure $\leftarrow \{e,p,c,l,i\}$

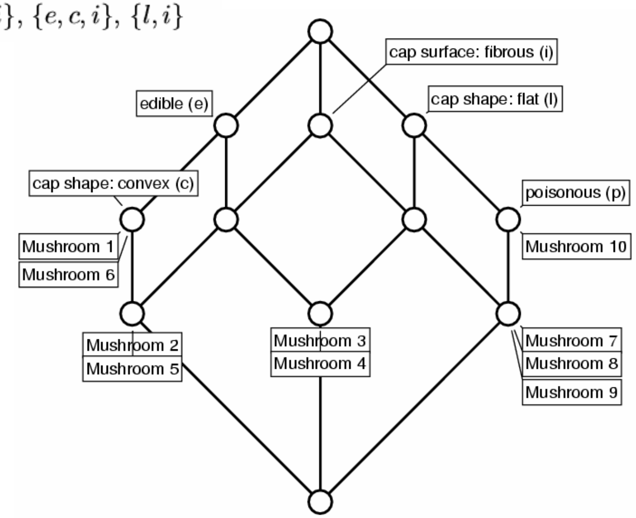
	edible (e)	poisonous (p)	cap shape: convex (c)	cap shape: flat (l)	cap surface: fibrous (i)
Mushroom 1	×		×		
Mushroom 2	×		×		×
Mushroom 3	×			×	×
Mushroom 4	×			×	×
Mushroom 5	×		×		×
Mushroom 6	×		×		
Mushroom 7		×		×	×
Mushroom 8		×		×	×
Mushroom 9		×		×	×
Mushroom 10		×		×	

Finally the algorithm collects all concept intents (step 14):

$$\emptyset,\ \{e\},\ \{p,l\},\ \{c,e\},\ \{l\},\ \{i\},\ \{e,p,c,l,i\},$$
$$\{e,l,i\},\ \{e,i\},\ \{p,l,i\},\ \{e,c,i\},\ \{l,i\}$$

(which are exactly the intents of the concepts of the concept lattice in Figure 8). The algorithm determined the support of $5 + 10 + 3 = 18$ attribute sets in three passes of the database.

$$\emptyset,\ \{e\},\ \{p,l\},\ \{c,e\},\ \{l\},\ \{i\},\ \{e,p,c,l,i\},$$
$$\{e,l,i\},\ \{e,i\},\ \{p,l,i\},\ \{e,c,i\},\ \{l,i\}$$



TITANIC vs. Next-Closure

- Next-Closure needs almost no memory.
- Next-Closure can exploit known symmetries between attributes.
- Next-Closure can be used for knowledge acquisition.
- TITANIC has far better performance, especially on large data sets.