Formal Concept Analysis: Themes and Variations for Knowledge Discovery

Sergei O. Kuznetsov\textsuperscript{1} and Amedeo Napoli\textsuperscript{2}
\textsuperscript{1} School of Applied Mathematics and Information Science, National Research University Higher School of Economics, Moscow, Russia
\textsuperscript{2} LORIA (CNRS – INRIA Nancy Grand-Est – Université de Lorraine)
B.P. 239, 54506 Vandoeuvre les Nancy, France
skuznetsov@yandex.ru; Amedeo.Napoli@loria.fr

Tutorial on Formal Concept Analysis at IJCAI 2013

IJCAI 2013, Beijing, August 3rd 2013
Summary of the presentation

Introduction

A Smooth Introduction to Formal Concept Analysis
  Derivation operators, formal concepts and concept lattice
  The structure of the concept lattice
  Scaling
  Two algorithms for extracting the concepts and building the concept lattice

Relational Concept Analysis

Pattern Structures

Conclusion and References
Introduction

A Smooth Introduction to Formal Concept Analysis
Derivation operators, formal concepts and concept lattice
The structure of the concept lattice
Scaling
Two algorithms for extracting the concepts and building the concept lattice

Relational Concept Analysis

Pattern Structures

Conclusion and References
The process of **Knowledge Discovery guided by Domain Knowledge (KDDK)** is applied on large volumes of data for extracting information units which are useful, significant, and reusable.

KDDK is based on four main operations: **data preparation**, **data mining**, **interpretation** and **representation** of the extracted units.

KDDK is **iterative and interactive**, guided by an **analyst**, and by **domain knowledge**.

KDDK is an interactive and iterative process that can be replayed.
One the core idea of KDDK is **classification**, which is involved in all tasks of **data and knowledge processing**:

- **mining**: Formal Concept Analysis (FCA), pattern mining...
- **modeling**: hierarchy of concepts and relations,
- **representing**: concepts and relations as knowledge units,
- **reasoning and problem solving**: classification-based and case-based reasoning.

KDDK is an interactive and iterative process that can be replayed.
KDDK is used for knowledge engineering and problem-solving activities in some application domains:

- agronomy
- astronomy
- biology
- chemistry
- cooking
- medicine
Introduction

A Smooth Introduction to Formal Concept Analysis
Derivation operators, formal concepts and concept lattice
The structure of the concept lattice
Scaling
Two algorithms for extracting the concepts and building the concept lattice

Relational Concept Analysis

Pattern Structures

Conclusion and References
FCA, Formal Concepts and Concept Lattices

The FCA process

- The basic procedure of Formal Concept Analysis (FCA) is based on a simple representation of data, i.e. a binary table called a formal context.
- Each formal context is transformed into a mathematical structure called concept lattice.
- The information contained in the formal context is preserved.
- The concept lattice is the basis for data analysis. It is represented graphically to support analysis, mining, visualization, interpretation...
The notion of a formal context

<table>
<thead>
<tr>
<th>Objects / Attributes</th>
<th>m1</th>
<th>m2</th>
<th>m3</th>
<th>m4</th>
<th>m5</th>
</tr>
</thead>
<tbody>
<tr>
<td>g1</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>g2</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>g3</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>g4</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>g5</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>g6</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
</tbody>
</table>

\[(G, M, I)\] is called a formal context where \(G\) (\emph{Gegenstände}) and \(M\) (\emph{Merkmale}) are sets, and \(I \subseteq G \times M\) is a binary relation between \(G\) and \(M\).

The elements of \(G\) are the objects, while the elements of \(M\) are the attributes, \(I\) is the incidence relation of the context \((G, M, I)\).
Two derivation operators

- For \( A \subseteq G \): \( A' = \{ m \in M / (g, m) \in I \text{ for all } g \in A \} \)
- Dually, for \( B \subseteq M \): \( B' = \{ g \in G / (g, m) \in I \text{ for all } m \in B \} \)

\( \{g3\}' \) and \( \{m3\}' \):

<table>
<thead>
<tr>
<th>Objects / Attributes</th>
<th>m1</th>
<th>m2</th>
<th>m3</th>
<th>m4</th>
<th>m5</th>
</tr>
</thead>
<tbody>
<tr>
<td>g1</td>
<td></td>
<td>( \times )</td>
<td>( \times )</td>
<td></td>
<td>( \times )</td>
</tr>
<tr>
<td>g2</td>
<td>( \times )</td>
<td></td>
<td>( \times )</td>
<td>( \times )</td>
<td></td>
</tr>
<tr>
<td>g3</td>
<td>( \times )</td>
<td>( \times )</td>
<td>( \times )</td>
<td>( \times )</td>
<td></td>
</tr>
<tr>
<td>g4</td>
<td>( \times )</td>
<td></td>
<td></td>
<td></td>
<td>( \times )</td>
</tr>
<tr>
<td>g5</td>
<td>( \times )</td>
<td>( \times )</td>
<td></td>
<td>( \times )</td>
<td></td>
</tr>
<tr>
<td>g6</td>
<td>( \times )</td>
<td></td>
<td></td>
<td>( \times )</td>
<td>( \times )</td>
</tr>
</tbody>
</table>
Two derivation operators

\[ \{g_3, g_5\}' \]

<table>
<thead>
<tr>
<th>Objects / Attributes</th>
<th>m1</th>
<th>m2</th>
<th>m3</th>
<th>m4</th>
<th>m5</th>
</tr>
</thead>
<tbody>
<tr>
<td>g1</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>g2</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>g3</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>g4</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>g5</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>g6</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>
Two derivation operators

\{m3, m4\}'

<table>
<thead>
<tr>
<th>Objects / Attributes</th>
<th>m1</th>
<th>m2</th>
<th>m3</th>
<th>m4</th>
<th>m5</th>
</tr>
</thead>
<tbody>
<tr>
<td>g1</td>
<td></td>
<td>×</td>
<td>×</td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>g2</td>
<td>×</td>
<td></td>
<td>×</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>g3</td>
<td>×</td>
<td>×</td>
<td></td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>g4</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>g5</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>g6</td>
<td>×</td>
<td></td>
<td>×</td>
<td>×</td>
<td></td>
</tr>
</tbody>
</table>
The derivation operators and the Galois connection

- The derivation operators establish a **Galois connection** between the power sets $\mathcal{P}(G)$ and $\mathcal{P}(M)$ (and thereby a dual isomorphism between two closure systems).

A Galois connection is defined as follows:

- Let $P$ and $Q$ be ordered sets.
  A pair of maps $\phi : P \rightarrow Q$ and $\psi : Q \rightarrow P$ is called a **Galois connection** between $P$ and $Q$ if:

  1. $p_1 \leq p_2 \implies \phi(p_1) \geq \phi(p_2)$
  2. $q_1 \leq q_2 \implies \psi(q_1) \geq \psi(q_2)$
  3. $p \leq \psi \circ \phi(p)$ and $q \leq \phi \circ \psi(q)$
The Galois connection and the closure operators

- $' : \wp(G) \rightarrow \wp(M)$ with $A \rightarrow A'$
- $' : \wp(M) \rightarrow \wp(G)$ with $B \rightarrow B'$
- These two applications induce a Galois connection between $\wp(G)$ and $\wp(M)$ when sets are ordered by set inclusion relation.
The Galois connection and the closure operators

A closure operator on a set $H$ is a map $\kappa$ such that:

- $\kappa : \wp(H) \rightarrow \wp(H)$
- For all $A_1, A_2 \subseteq H$:
  - (i) $A_1 \subseteq \kappa(A_1)$
  - (ii) $A_1 \subseteq A_2$ then $\kappa(A_1) \subseteq \kappa(A_2)$
  - (iii) $\kappa(\kappa(A_1)) = \kappa(A_1)$
- A is a closed set whenever $\kappa(A) = A$.
- The composition operators $\circ$ (composition of $'$ and $'$) are closure operators.
Given a formal context \((G, M, I)\):

- \(A' = \{m \in M / (g, m) \in I \text{ for all } g \in A\}\)
- \(B' = \{g \in G / (g, m) \in I \text{ for all } m \in B\}\)
- \((A, B)\) is a formal concept of \((G, M, I)\) iff: \(A \subseteq G, B \subseteq M, A' = B, \text{ and } A = B'\).
- \(A\) is the extent and \(B\) is the intent of \((A, B)\).
- The mappings \(A \rightarrow A''\) and \(B \rightarrow B''\) are closure operators.
The concept lattice

- Formal concepts can be ordered by:
  
  \[(A_1, B_1) \leq (A_2, B_2) \iff A_1 \subseteq A_2 \text{ (dually } B_2 \subseteq B_1)\].

- The set \(\mathcal{B}(G, M, I)\) of all formal concepts of \((G, M, I)\) with this order is a complete lattice called the concept lattice of \((G, M, I)\).

- Recall that: a set \((P, \leq)\) is a complete lattice if the supremum \(\bigvee S\) and the infimum \(\bigwedge S\) exist for any subset \(S\) of \(P\).

- Every complete lattice has a top or unit element denoted by \(\top\), and a bottom or zero element denoted by \(\bot\).
The concept lattice

<table>
<thead>
<tr>
<th>G / M</th>
<th>m1</th>
<th>m2</th>
<th>m3</th>
<th>m4</th>
<th>m5</th>
</tr>
</thead>
<tbody>
<tr>
<td>g1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>g2</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>g3</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>g4</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>g5</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>g6</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
</tbody>
</table>

Diagram:

0

\( l = \{\} \)
\( E = \{g_1, g_2, g_3, g_4, g_5, g_6\} \)

1

\( l = \{m_3\} \)
\( E = \{g_1, g_2, g_3, g_5, g_6\} \)

2

\( l = \{m_1, m_4\} \)
\( E = \{g_2, g_3, g_4, g_5, g_6\} \)

3

\( l = \{m_2, m_3\} \)
\( E = \{g_1, g_3, g_5\} \)

4

\( l = \{m_1, m_3, m_4\} \)
\( E = \{g_2, g_3, g_5, g_6\} \)

5

\( l = \{m_2, m_3, m_5\} \)
\( E = \{g_1\} \)

6

\( l = \{m_1, m_2, m_3, m_4\} \)
\( E = \{g_3, g_5\} \)

7

\( l = \{m_1, m_2, m_3, m_4, m_5\} \)
\( E = \{\} \)
The basic theorem of FCA

- The concept lattice \( \mathfrak{B}(G, M, I) \) is a complete lattice in which the infimum and the supremum are given by:
  
  \[ \bigwedge_{k \in K} (A_k, B_k) = (\bigcap_{k \in K} A_k, (\bigcup_{k \in K} B_k)'' \) \]
  
  \[ \bigvee_{k \in K} (A_k, B_k) = ((\bigcup_{k \in K} A_k)'', \bigcap_{k \in K} B_k) \]

- **Note:** an intersection of closed sets is a closed set but a union of closed sets is not necessarily a closed set.
The properties of the derivation operators
Iterating derivation

- $A' = \{ m \in M / (g, m) \in I \text{ for all } g \in A \}$
- $B' = \{ g \in G / (g, m) \in I \text{ for all } m \in B \}$
- The derivation operators can be combined: Starting with a set $A \subseteq G$, we obtain that $A'$ is a subset of $M$.
- Applying the second operator on this set, we get $(A')'$, or $A''$ for short, which is a set of objects.
- Continuing, we obtain $A''', A'''', \text{ and so on.}$
Properties of the derivation operators

- $A' = \{ m \in M | (g, m) \in I \text{ for all } g \in A \}$
- $B' = \{ g \in G | (g, m) \in I \text{ for all } m \in B \}$

The derivation operator $'$ satisfy the following rules:

- $A_1 \subseteq A_2 \implies A_2' \subseteq A_1'$
- $B_1 \subseteq B_2 \implies B_2' \subseteq B_1'$
- $A \subseteq A'' \text{ and } A' = A'''$
- $B \subseteq B'' \text{ and } B' = B'''$
Examples

<table>
<thead>
<tr>
<th>G / M</th>
<th>m1</th>
<th>m2</th>
<th>m3</th>
<th>m4</th>
<th>m5</th>
</tr>
</thead>
<tbody>
<tr>
<td>g1</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>g2</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>g3</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>g4</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>g5</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>g6</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>

\[ A_1 \subseteq A_2 \implies A'_2 \subseteq A'_1 \]
\[ B_1 \subseteq B_2 \implies B'_2 \subseteq B'_1 \]
\[ A \subseteq A'' \text{ and } A' = A''' \]
\[ B \subseteq B'' \text{ and } B' = B''' \]
The derivation operator \(^\prime\) satisfy the following rules:

- Combining the derivation operators, we get two operators of the form: \(X \rightarrow X''\), one on \(G\), the other on \(M\).
- For \(A \subseteq G\) we have that \(A'' \subseteq G\).
  The set \(A''\) is called the extent closure of \(A\).
- Dually, when \(B \subseteq M\) we have also that \(B'' \subseteq M\).
  The set \(B''\) is called the intent closure of \(A\).
Extent closure, intent closure

From $A \subseteq A''$ and $B \subseteq B''$ it comes:

- (i) whenever all objects from a set $A \subseteq G$ have a common attribute $m$, then also all objects from $A''$ have that attribute.
- (ii) whenever an object $g \in G$ has all attributes from $B \subseteq M$, then this object also has all attributes from $B''$. 
For $A_1, A_2 \subseteq G$, and dually for $B_1, B_2 \subseteq M$, we have:

- $A_1 \subseteq A_2 \implies A'_1 \subseteq A''_2$
- $B_1 \subseteq B_2 \implies B'_1 \subseteq B''_2$
- $(A'')'' = A''$
- $(B'')'' = B''$
Closure operators

- As already mentioned, the operators ‘ and ″ satisfying the above properties are closure operators.
- The sets which are images of a closure operator are the closed sets.
- Thus, in the case of a closure operator \( X \rightarrow X'' \) the closed sets are the sets of the form \( X'' \).
Closed sets are intents and extents

- If \((G, M, I)\) is a formal context and \(A \subseteq G\), then \(A''\) is an extent.
- Conversely, if \(A\) is an extent of \((G, M, I)\), then \(A = A''\).
- Dually if \(B\) is an intent of \((G, M, I)\), then \(B = B''\), and every intent \(B\) satisfies \(B = B''\).
- This follows from the fact that for each subset \(A \subseteq G\), the pair \((A'', A')\) is a formal concept, and that similarly, for each subset \(B \subseteq M\), \((B', B'')\) is a formal concept.
- Therefore, the closed sets of the closure operator \(A \rightarrow A''\), \(A \subseteq G\) are precisely the extents of \((G, M, I)\), and the closed sets of the operator \(B \rightarrow B''\), \(B \subseteq M\), are precisely the intents.
The structure of the concept lattice
The reduced labeling

- A reduced labeling may be used allowing that each object and each attribute is entered only once in a diagram.
- The name of the object $g$ is attached to the “lower half” of the corresponding object concept $\gamma(g) = (\{g\}'', \{g\}')$.
- The object concept of an object $g \in G$ is the concept $(\{g\}'', \{g\}')$ where $\{g\}'$ is the object intent $\{m \in M / gI_m\}$ of $g$.
- The object concept of $g$, denoted by $\gamma(g)$, is the smallest concept (for the lattice order) with $g$ in its extent.

Example:
\[
\begin{align*}
\gamma(g_4) &= (\{g_4\}'', \{g_4\}') = (\{g_2, g_3, g_4, g_5, g_6\}, \{m_1, m_4\}) \\
\gamma(g_1) &= (\{g_1\}'', \{g_1\}') = (\{g_1\}, \{m_2, m_3, m_5\})
\end{align*}
\]
The reduced labeling

- The name of the attribute $m$ is located to the “upper half” of the corresponding attribute concept $\mu(m) = (\{m\}', \{m\}'')$.

- Correspondingly, the attribute concept of an attribute $m \in M$ is the concept $((\{m\}', \{m\}''))$ where $\{m\}'$ is the attribute extent $\{g \in G/gIm\}$ of $m$.

- The attribute concept of $m$, denoted by $\mu(m)$ is the largest concept (for the lattice order) with $m$ in its intent.

- Example:
\[
\mu(m_1) = (\{m_1\}', \{m_1\}'') = (\{g_2, g_3, g_4, g_5, g_6\}, \{m_1, m_4\})
\]
\[
\mu(m_1) = \mu(m_4)
\]
\[
\mu(m_2) = (\{m_2\}', \{m_2\}'') = (\{g_1, g_3, g_5\}, \{m_2, m_3\})
\]
The reduced labeling

<table>
<thead>
<tr>
<th>G / M</th>
<th>m1</th>
<th>m2</th>
<th>m3</th>
<th>m4</th>
<th>m5</th>
</tr>
</thead>
<tbody>
<tr>
<td>g1</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>g2</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>g3</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>g4</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>g5</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>g6</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
</tbody>
</table>

Reduced labeling: The attributes “at the highest” and the objects “at the lowest”.
The reduced labeling

- For any concept \((A, B)\) we have:
  - \(g \in A \iff \gamma(g) \leq (A, B)\)
  - \(m \in B \iff (A, B) \leq \mu(m)\)
An extent is an ideal (down-set)

- The extent of an arbitrary concept can be found as the set of objects in the principal ideal generated by the concept.
- Let \((P, \leq)\) be an ordered set. A subset \(Q \subseteq P\) is an order ideal or a down-set if \(x \in Q\) and \(y \leq x\) imply that \(y \in Q\).
- \(\downarrow Q = \{y \in P/\exists x \in Q : y \leq x\}\)
- \(\downarrow x = \{y \in P/y \leq x\}\)
An intent is a filter (up-set)

- The intent of an arbitrary concept can be found as the set of objects in the principal filter generated by the concept.
- Let \((P, \leq)\) be an ordered set. A subset \(Q \subseteq P\) is an order filter or an up-set if \(x \in Q\) and \(x \leq y\) imply that \(y \in Q\).
- \(\uparrow Q = \{y \in P / \exists x \in Q : x \leq y\}\)
- \(\uparrow x = \{y \in P / x \leq y\}\)
Given a concept lattice, the context associated to the lattice can be read using the following general rule:
\((g, m) \in I \iff \gamma(g) \leq \mu(m)\)

Just as a set of concepts can be uniquely determined from a given context, so the context can be reconstructed from its concepts.

The set \(G\) is the extent of the largest concept \((G, G')\).

The set \(M\) is the intent of the smallest concept \((M', M)\).

The incidence relation is given by:
\[ I = \bigcup \{ A \times B / (A, B) \in C(G, M, I) \} \]
where \(C(G, M, I)\) denotes the set of all formal concepts for \((G, M, I)\).
Types of attributes

- **Introducing an attribute**: an attribute $\alpha$ is introduced in a concept $C$ when it is not present in any ascendant (super-concept) of $C$, i.e. the concept $C$ corresponds to the attribute concept of $\alpha$ (sometimes called the introducer of $\alpha$).

- **Inheriting an attribute**: an attribute $\alpha$ is inherited by a concept $C$ when it is already present in an ascendant of $C$, i.e. $C$ is lower for the lattice order than the attribute-concept or introducer of $\alpha$. 
Types of attributes (example)

- \(m_3\) is an attribute introduced in the concept \((g_{12356}, m_3)\), \(a\) and \(d\) are attributes introduced in the concept \((g_{23456}, m_{14})\), \(b\) is an attribute introduced in the concept \((g_{135}, m_{23})\).

- \(m_3\) is an attribute inherited by \((g_{135}, m_{23})\), \(m_1\), \(m_3\), and \(m_4\), are attributes inherited by \((g_{2356}, m_{134})\), and so on.
Extracting rules from a concept lattice

Mutual implications between attributes having the same attribute-concept

- Attributes having the same attribute-concept or introducer are equivalent: for example \( m1 \leftrightarrow m4 \) for \((g23456, m14)\).
Extracting rules from a concept lattice (continued)

Introduced attributes imply inherited attributes

- When an attribute $\alpha$ is introduced, it implies every inherited attribute in the attribute-concept of $\alpha$: for example $m_2 \rightarrow m_3$ for $(g_{135}, m_{23})$ and $m_5 \rightarrow m_{23}$ for $(g_1, m_{235})$. 
Scaling
When the size of the data set is growing, it becomes unreasonable to display the full data in a single lattice diagram. Formal Concept Analysis allows to:

- split large diagrams into smaller ones, so that the information content is preserved,
- browse through lattices and thereby build conceptual views of data.
The need for scaling

- There are many series of formal contexts that have an suggestive interpretation. Such formal contexts will be called scales.
- So formally, a scale is the same as a formal context. But it is meant to have a special interpretation.
- Examples of scales are nominal, ordinal, and dichotomic scales.
Conceptual scaling

- The formal context is the basic data type of Formal Concept Analysis.
- However data are often given in form of a many-valued context.
- Many-valued contexts are translated to one-valued context via conceptual scaling.
- But this is not automatic and some arbitrary choices have to be made.
The example of the context of planets

<table>
<thead>
<tr>
<th>Planet</th>
<th>Size</th>
<th>Distance to Sun</th>
<th>Moon(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jupiter</td>
<td>large</td>
<td>far</td>
<td>yes</td>
</tr>
<tr>
<td>Mars</td>
<td>small</td>
<td>near</td>
<td>yes</td>
</tr>
<tr>
<td>Mercury</td>
<td>small</td>
<td>near</td>
<td>no</td>
</tr>
<tr>
<td>Neptune</td>
<td>medium</td>
<td>far</td>
<td>yes</td>
</tr>
<tr>
<td>Pluto</td>
<td>small</td>
<td>far</td>
<td>yes</td>
</tr>
<tr>
<td>Saturn</td>
<td>large</td>
<td>far</td>
<td>yes</td>
</tr>
<tr>
<td>Earth</td>
<td>small</td>
<td>near</td>
<td>yes</td>
</tr>
<tr>
<td>Uranus</td>
<td>medium</td>
<td>far</td>
<td>yes</td>
</tr>
<tr>
<td>Venus</td>
<td>small</td>
<td>near</td>
<td>no</td>
</tr>
</tbody>
</table>
## The context of planets after nominal scaling

<table>
<thead>
<tr>
<th>Planet</th>
<th>Size</th>
<th>Distance to Sun</th>
<th>Moon(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>small</td>
<td>near</td>
<td>yes</td>
</tr>
<tr>
<td>Jupiter</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Mars</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Mercury</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Neptune</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Pluto</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Saturn</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Earth</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Uranus</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Venus</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>
The concept lattice of planets (after scaling)
Examples of scaling

- **Nominal**: $K = (N, N, =)$
- **Ordinal**: $K = (N, N, \leq)$
- **Interordinal**: $K = (N, N, \leq \cup \geq)$
Two algorithms for building formal concepts and the concept lattice
A rectangle in a binary table corresponds to a pair \((X, Y)\) –where \(X\) denotes an extension and \(Y\) denotes an intension– only contains crosses \(x\). Such an extension and intension are not necessarily extents and intents respectively.

A rectangle \((X, Y)\) is contained in another rectangle \((X_1, Y_1)\) whenever \(X \subseteq X_1\) and \(Y \subseteq Y_1\).

A rectangle \((X, Y)\) is maximal when it is not included in any other rectangle: any rectangle \((X_1, Y_1)\) containing a maximal rectangle \((X, Y)\) is such that \(X_1\) and/or \(Y_1\) contain at least a “void place”, i.e. a place without a cross \(x\).
An algorithm for constructing the concept lattice

- **Step 1**: build the rectangles \((X, Y)\) whose extension \(X\) is of size 1 (the cardinality of \(X\) is equal to 1).

- **Step 2 with union of rectangles**: build the rectangles \((X, Y)\) of size 2 making the union of extensions of size 1 and the intersection of intensions. An intersection should not be empty otherwise the corresponding rectangle is no more considered.

- **Step 3**: Check and remove the rectangles that are not maximal, then assemble rectangles with the same intension.

- **Step 4**: continue in the same way the process considering rectangles of size 3, 4, and so on...

- **Final step**: the building process stops as soon as there is no more rectangle to be built.
An example of construction of a concept lattice (1)

<table>
<thead>
<tr>
<th>G / M</th>
<th>m1 (a)</th>
<th>m2 (b)</th>
<th>m3 (c)</th>
<th>m4 (d)</th>
<th>m5 (e)</th>
</tr>
</thead>
<tbody>
<tr>
<td>g1</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>g2</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>g3</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>g4</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>g5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>g6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>

For better readability: \( M = \{a, b, c, d, e\} \)

The rectangles of size 1:
\{g1\} \times \{b, c, e\}, \{g2\} \times \{a, c, d\}, \{g3\} \times \{a, b, c, d\}, \{g4\} \times \{a, d\}, \{g5\} \times \{a, b, c, d\}, \{g6\} \times \{a, c, d\}
The rectangles of size 1:
\{g1\} \times \{b,c,e\}, \{g2\} \times \{a,c,d\}, \{g3\} \times \{a,b,c,d\}, \{g4\} \times \{a,d\},
\{g5\} \times \{a,b,c,d\}, \{g6\} \times \{a,c,d\}

Build the rectangles of size 2 by union of rectangles of size 1:
\{g1,g2\} \times \{c\}, \{g1,g3\} \times \{b,c\}, \{g1,g5\} \times \{b,c\}, \{g1,g6\} \times \{c\},
\{g2,g3\} \times \{a,c,d\}, \{g2,g4\} \times \{a,d\}, \{g2,g5\} \times \{a,c,d\}, \{g2,g6\} \times \{a,c,d\},
\{g3,g4\} \times \{a,d\}, \{g3,g5\} \times \{a,b,c,d\}, \{g3,g6\} \times \{a,c,d\},
\{g4,g5\} \times \{a,d\}, \{g4,g6\} \times \{a,d\},
\{g5,g6\} \times \{a,c,d\}, ...
Remove the non maximal rectangles
(the rectangles with the same intension and a smaller extension):
For example:
\{g2\}x\{a,c,d\} because of \{g2,g3\}x\{a,c,d\},
\{g3\}x\{a,b,c,d\} and \{g5\}x\{a,b,c,d\} because of \{g3,g5\}x\{a,b,c,d\},
\{g4\}x\{a,d\} because of \{g4,g5\}x\{a,d\},
\{g6\}x\{a,c,d\} because of \{g5,g6\}x\{a,c,d\}
...
Fusion of rectangles with the same intension:

For example:

\{g1,g2\} \times \{c\} \text{ and } \{g1,g6\} \times \{c\} \text{ give } \{g1,g2,g6\} \times \{c\}

\{g1,g3\} \times \{b,c\} \text{ and } \{g1,g5\} \times \{b,c\} \text{ give } \{g1,g3,g5\} \times \{b,c\}

\{g2,g3\} \times \{a,c,d\}, \{g2,g5\} \times \{a,c,d\} \text{ and } \{g2,g6\} \times \{a,c,d\} \text{ give } \\
\{g2,g3,g5,g6\} \times \{a,c,d\}

This removes the fusion: \{g3,g6\} \times \{a,c,d\} \text{ and } \{g5,g6\} \times \{a,c,d\} \text{ give } \\
\{g3,g5,g6\} \times \{a,c,d\}

\{g2,g4\} \times \{a,d\}, \{g3,g4\} \times \{a,d\}, \{g4,g5\} \times \{a,d\} \text{ and } \{g4,g6\} \times \{a,d\} \text{ give } \\
\{g2,g3,g4,g5,g6\} \times \{a,d\}
An example of construction of a concept lattice (5)

Listing the rectangles by size:
Size 1: \{g1\} \times \{b,c,e\},
Size 2: \{g3,g5\} \times \{a,b,c,d\},
Size 3: \{g1,g3,g5\} \times \{b,c\}, \{g1,g2,g6\} \times \{c\},
Size 4: \{g2,g3,g5,g6\} \times \{a,c,d\},
Size 5: \{g2,g3,g4,g5,g6\} \times \{a,d\}

A last construction by union of rectangles is possible:
\{g1,g3,g5\} \times \{b,c\} and \{g1,g2,g6\} \times \{c\} give \{g1,g2,g3,g5,g6\} \times \{c\},
this removes \{g1,g2,g6\} \times \{c\}
An example of construction of a concept lattice (6)
The Bordat algorithm

Let us consider a set of objects $G$ and a set of attributes $M$.

- The **Bordat algorithm** is able to build the formal concepts and the order between the concepts, thus the whole concept lattice.
- Let us consider a set of objects $G$ and a set of attributes $M$.
- Set the **bottom concept** as the pair $\bot = (M', M)$ unless there is one object owning all attributes (then $M' \neq \emptyset$).
The Bordat algorithm

- Let $C_k = (X_k, Y_k)$ be a formal concept.
- The concept $C_{k+1} = (X_{k+1}, Y_{k+1})$ is a subsumer of $C_k$ in the lattice whenever it verifies:
  - (i) $X_{k+1} = X_k \cup \{x \in G \setminus X_k | f_{C_k}(x) = Y_{k+1}\}$ where $f_{C_k}$ computes the intension of $x$ restricted to $(G \setminus X_k) \times Y_k$ (extensions are growing).
  - (ii) $Y_{k+1} \in \text{Max}\{f_{C_k}(x) | x \not\in X_k\}$
The Bordat algorithm

\[
\begin{array}{|c|ccc|c|}
\hline
G / M & a & b & c & d & e \\
\hline
g1 & x & x & & x \\
g2 & x & x & & \\
g3 & x & x & &
\hline
g4 & x & & &
\hline
g5 & x & x & &
\hline
g6 & x & x & &
\hline
\end{array}
\]

\[G = \{g_1, g_2, g_3, g_4, g_5, g_6\}\]
and \[M = \{a, b, c, d, e\}\]

\[\perp = (\emptyset, M) = (X_0, Y_0)\]
The Bordat algorithm

- \( G = \{g1, g2, g3, g4, g5, g6\} \) and \( M = \{a, b, c, d, e\} \)
- \( \perp = (\emptyset, M) = (X_0, Y_0) \)
- \( X_1 = X_0 \cup \{x \in G \setminus X_0 | f_{C_0}(x) = Y_1\} \)
- \( Y_1 \in \text{Max}\{f_{C_0}(x)|x \not\in X_0\} \)
- \( f_{C_0} : (G \setminus \emptyset) \times \{a, b, c, d, e\} \)
- \( \{f_{C_0}(x)|x \not\in X_0\} = \{f_{C_0}(x)|x \in G\} = \{f_{C_0}(g1), f_{C_0}(g2), f_{C_0}(g3), f_{C_0}(g4), f_{C_0}(g5), f_{C_0}(g6)\} = \{\{b, c, e\}, \{a, c, d\}, \{a, b, c, d\}, \{a, d\}, \{a, b, c, d\}, \{a, c, d\}\} \)
The Bordat algorithm

- \[\text{Max}\{\{b, c, e\}, \{a, c, d\},\{a, b, c, d\},\{a, d\},\{a, b, c, d\},\{a, c, d\}\} = \{\{b, c, e\}, \{a, b, c, d\}\}\]
- \[Y_1 = \{b, c, e\}\]
- \[X_1 = X_0 \cup \{x \in G \setminus X_0 | f_{C_0}(x) = Y_1\} = \{x \in G | f_{C_0}(x) = \{b, c, e\}\} = \{1\}\]
- \[Y_2 = \{a, b, c, d\}\]
- \[X_2 = X_0 \cup \{x \in G \setminus X_0 | f_{C_0}(x) = Y_2\} = \{x \in G | f_{C_0}(x) = \{a, b, c, d\}\} = \{3, 5\}\]
- \[C_1 = (X_1, Y_1) = (\{g_1\}, \{b, c, e\})\] and \[C_2 = (X_2, Y_2) = (\{g_3, g_5\}, \{a, b, c, d\})\] are the direct subsumers of \((X_0, Y_0)\) in the lattice.
The Bordat algorithm

- $X_3 = X_1 \cup \{x \in G \setminus X_1 | f_{c_1}(x) = Y_3\}$
- $f_{c_1} : (G \setminus \{g1\}) \times \{b, c, e\}$
- $Y_3 \in \text{Max}\{f_{c_1}(x) | x \in \{g2, g3, g4, g5, g6\}\}$
- $Y_3 \in \text{Max}\{f_{c_1}(g2), f_{c_1}(g2), f_{c_1}(g3)f_{c_1}(g4), f_{c_1}(g5), f_{c_1}(g6)\}$
  $= \text{Max}\{\{c\}, \{b, c\}, \emptyset, \{b, c\}, \{c\}\}$
- Thus $C_3 = (X_3, Y_3) = (\{g1, g3, g5\}, \{b, c\})$
The Bordat algorithm

- $X_4 = X_2 \cup \{x \in G \setminus X_2 | f_{C_2}(x) = Y_4\}$
- $f_{C_2} : (G \setminus \{g3, g5\}) \times \{a, b, c, d\}$
- $Y_4 \in \text{Max}\{f_{C_2}(x)|x \in \{g1, g2, g4, g6\}\}$
- $Y_4 \in \text{Max}\{f_{C_2}(g1), f_{C_2}(g2), f_{C_2}(g4), f_{C_2}(g6)\}$
  - $= \text{Max}\{\{b, c\}, \{a, c, d\}, \{a, d\}, \{a, c, d\}\}$
  - $= \{\{b, c\}, \{a, c, d\}\}$
- Thus the first subsumer is identical to
  $C_3 = (X_3, Y_3) = (\{g1, g3, g5\}, \{b, c\})$
- The second subsumer
  $C_4 = (X_4, Y_4) = (\{g2, g3, g5, g6\}, \{a, c, d\})$
The Bordat algorithm

- $X_5 = X_3 \cup \{x \in G \setminus X_3 | f_{c_3}(x) = Y_5\}$
- $f_{c_3} : (G \setminus \{g_1, g_3, g_5\}) \times \{b, c\}$
- $Y_5 \in \text{Max}\{f_{c_3}(x) | x \in \{g_2, g_4, g_6\}\}$
- $Y_5 \in \text{Max}\{f_{c_3}(g_2), f_{c_3}(g_4), f_{c_3}(g_6)\}$
  \[= \text{Max}\{\{c\}, \emptyset, \{c\}\} = \{c\}\]
- Thus $C_5 = (X_5, Y_5) = (\{g_1, g_2, g_3, g_5, g_6\}, \{c\})$
The Bordat algorithm

- $X_6 = X_4 \cup \{x \in G \setminus X_4 | f_{C_4}(x) = Y_6\}$

- $f_{C_4} : (G \setminus \{g_2, g_3, g_5, g_6\}) \times \{a, c, d\}$

- $Y_6 \in \text{Max}\{f_{C_4}(x) | x \in \{g_1, g_4\}\}$

- $Y_6 \in \text{Max}\{f_{C_4}(g_1), f_{C_4}(g_4)\} = \text{Max}\{\{c\}, \{d\}\}$

- The first subsumer is identical to
  $C_5 = (X_5, Y_5) = (\{g_1, g_2, g_3, g_5, g_6\}, \{c\})$

- The second subsumer
  $C_6 = (X_6, Y_6) = (\{g_2, g_3, g_4, g_5, g_6\}, \{d\})$

- ...
Introduction

A Smooth Introduction to Formal Concept Analysis
  Derivation operators, formal concepts and concept lattice
  The structure of the concept lattice
Scaling
  Two algorithms for extracting the concepts and building the concept lattice

Relational Concept Analysis

Pattern Structures

Conclusion and References
Relational Concept Analysis


Introducing Relational Concept Analysis (RCA)

- The objective of RCA is to extend the purpose of FCA for taking into account relations between objects.
- The RCA process relies on the following main points:
  - a relational model based on the entity-relationship model,
  - a conceptual scaling process allowing to represent relations between objects as relational attributes,
  - an iterative process for designing a concept lattice where concept intents include binary and relational attributes.
- The RCA process provides “relational structures” that can be represented as ontology concepts within a knowledge representation formalism such as description logics (DLs).
The RCA data model

- The RCA data model relies on a so-called relational context family denoted by $\mathcal{RCF} = (\mathcal{K}, \mathcal{R})$, where:
- $\mathcal{K}$ is a set of formal contexts $\mathcal{K}_i = (G_i, M_i, I_i)$,
- $\mathcal{R}$ is a set of relations $r_k \subseteq G_i \times G_j$, where $G_i$ and $G_j$ are sets of objects from the formal contexts $\mathcal{K}_i$ and $\mathcal{K}_j$.
- A relation $r \subseteq G_i \times G_j$ has a domain and a range where:
  - $\text{dom}(r) = G_i$ and $\text{ran}(r) = G_j$. 
An example

- Suppose that we have a context $\text{Papers} \times \text{Topics}$ where Papers denotes a set of papers –from “a” to “ℓ”– and Topics denotes a set of three attributes, namely “lt” for “lattice theory”, “mmi” for “man-machine interface”, and “se” for “software engineering”.

- There are two relations:
  - $\text{cites} \subseteq \text{Papers} \times \text{Papers}$ indicates that a paper is citing another paper,
  - $\text{develops} \subseteq \text{Papers} \times \text{Papers}$ indicates that a paper is developing another paper.
The initial relational context

Relational context:

$$(K, R) = (\mathcal{K}_0, \{\text{cites, develops}\}) \text{ with } \mathcal{K}_0 = (\text{Papers, Topics, I})$$

<table>
<thead>
<tr>
<th></th>
<th>lt</th>
<th>mmi</th>
<th>se</th>
<th>a</th>
<th>b</th>
<th>g</th>
<th>h</th>
<th>c</th>
<th>d</th>
<th>i</th>
<th>j</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>g</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>h</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>j</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>k</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>
The $L_0$ concept lattice built from formal context $K_0$
Introducing relational scaling

- The first step consists in building an initial concept lattice $\mathcal{L}_0$ from the initial context $\mathcal{K}_0$ using standard FCA.
- The second step takes into account relations $r(o_i, o_j)$ for building a new context $\mathcal{K}_1$:
  - $r(o_i, o_j)$ means that object $o_i \in G_i$ is related through relation $r$ with object $o_j \in G_j$,
  - then a relational attribute of the form $\exists r.C_k$ is associated to object $o_i$ in $\mathcal{K}_1$, where $C_k$ is any concept instantiating $o_j$ in $\mathcal{L}_0$.
- When all relations between objects have been examined, the new context $\mathcal{K}_1$ is completed and a new concept lattice $\mathcal{L}_1$ is built accordingly.
Object $i$ is in relation with object $a$ through relation cites.

Object $a$ is in the extent of concepts $C_0$ and $C_2$ of the initial lattice $L_0$.

Thus, object $i$ is given two new relational attributes, namely $\exists \text{cites}:C_0$ and $\exists \text{cites}:C_2$.

Object $j$ is in relation with object $b$ through cites: by the same way, object $j$ is given two relational attributes $\exists \text{cites}:C_0$ and $\exists \text{cites}:C_2$. 
### The relational context $\mathcal{K}_0$

<table>
<thead>
<tr>
<th></th>
<th>lt</th>
<th>mmi</th>
<th>se</th>
<th>a</th>
<th>b</th>
<th>g</th>
<th>h</th>
<th>c</th>
<th>d</th>
<th>i</th>
<th>j</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>g</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>i</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>j</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>k</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>ℓ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Object $c$ is in relation with objects $a$ and $g$ through relation cites.

- Object $a$ is in the extent of concepts $C_0$ and $C_2$ in $L_0$ while object $g$ is in the extent of concepts $C_3$ and $C_2$ in $L_0$.
- Thus, object $c$ is given three new relational attributes, namely $\exists \text{cites}: C_0$, $\exists \text{cites}: C_2$, and $\exists \text{cites}: C_3$. 

\begin{itemize}
  \item Object $c$ is in relation with
\end{itemize}
Relational scaling in $L_0$

- The same process is applied to develops:
  - $e$ is in relation with $c$, $f$ with $d$, $k$ with $i$, and $\ell$ with $j$.
  - The four objects $e$, $f$, $k$, and $\ell$, are given the relational attribute $\exists$develops$:$C_2$. 

```

```
The new relational context $\mathcal{K}_1$

<table>
<thead>
<tr>
<th></th>
<th>lt</th>
<th>mmi</th>
<th>se</th>
<th>cites:c2</th>
<th>cites:c0</th>
<th>cites:c3</th>
<th>cites:c4</th>
<th>develops:c2</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>e</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>f</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>g</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>j</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>k</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>l</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

S.O. Kuznetsov and A. Napoli

Tutorial on FCA at IJCAI 2013
The concept lattice $\mathcal{L}_1$
Three forms of relational attributes

- **Existential scaling** $\exists r. C$: $r(o) \cap \text{Extent}(C) \neq \emptyset$
- **Universal scaling** $\forall r. C$: $r(o) \subseteq \text{Extent}(C)$
- **Universal-Existential scaling** $\forall \exists r. C$: $r(o) \subseteq \text{extent}(C)$ and $r(o) \neq \emptyset$

- With relational scaling, the homogeneity of concept descriptions is kept: all attributes – included relational attributes – are considered as binary attributes.
- **Standard algorithms** for building concept lattices can be straightforwardly reused.
Relational scaling in \( \mathcal{L}_1 \)

- The process is applied a second time for relation develops.
  - The object \( e \) develops \( c \) whose description has changed, i.e. \( c \) is in the extent of concepts \( C_2, C_5, \) and \( C_6 \).
  - Thus object \( e \) is given the relational attributes \( \exists \text{develops:} C_5 \) and \( \exists \text{develops:} C_6 \).
The new relational context $\mathcal{K}_2$

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>g</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>j</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>k</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>l</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

S.O. Kuznetsov and A. Napoli  Tutorial on FCA at IJCAI 2013
The concept lattice $\mathcal{L}_2$
The completion of the RCA process

- Relational scaling is still applied for cites and develops but the final context and the associated concept lattice are obtained after the second step.

- More generally, relational scaling is applied and either there are new modifications (continuation) or there is no more modification (fix-point).

- The relational scaling process reaches a fix-point and the final context is stationary: no more changes need to be made and the associated final lattice is reached (the relational scaling process terminates).
The concepts of the final concept lattice can be represented within a DL formalism such as $\mathcal{ALE}$ for designing an ontology schema supported by the lattice.

Some problems about knowledge representation are arising for representing binary and relational attributes. Binary attributes can be represented as atomic concepts.

Thanks to the semantics associated with relational scaling and operators, roles can be attached to defined concepts in a “natural” way using a construction such as $\exists r.C$. 

From a relational concept lattice to an ontology schema.
Text Mining
The text mining process

- **Text mining** implies the manipulation of textual documents in depth, i.e. w.r.t. their **structure** and their **content**.
- Text mining can be guided by **domain ontologies** and supports **ontology engineering**.
- In turn, ontologies can be used for **text annotation**, **information retrieval**, and for guiding the **mining of documents** w.r.t. their content.
From a relational concept lattice to an ontology schema
From a relational concept lattice to an ontology schema

Relational concept lattice:

$$C_8: \text{Intent} = \{\text{sticks}, \text{resist} : A_0, \text{resist} : A_1\}$$

$$C_8: \text{Extent} = \{\text{klebsiella}_P, \text{klebsiella}_O, \text{mycobacterium}_S\}$$

DL expressions:

$$C_8 \equiv \exists\text{sticks}.\top \sqcap \exists\text{resist}.A_0 \sqcap \exists\text{resist}.A_1$$

$$C_8(\text{klebsiella}_P), C_8(\text{klebsiella}_O), C_8(\text{mycobacterium}_S)$$
Introduction

A Smooth Introduction to Formal Concept Analysis
Derivation operators, formal concepts and concept lattice
The structure of the concept lattice
Scaling
Two algorithms for extracting the concepts and building the concept lattice

Relational Concept Analysis

Pattern Structures

Conclusion and References
Pattern Structures


Handling numerical data with FCA?

Conceptual scaling (discretization or binarization)

An object has an attribute if its value lies in a predefined interval

<table>
<thead>
<tr>
<th></th>
<th>$m_1$</th>
<th>$m_2$</th>
<th>$m_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$g_1$</td>
<td>5</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>$g_2$</td>
<td>6</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>$g_3$</td>
<td>4</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>$g_4$</td>
<td>4</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>$g_5$</td>
<td>5</td>
<td>8</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$m_1$, [4, 5]</th>
<th>$m_2$, [4, 7]</th>
<th>$m_3$, [5, 6]</th>
</tr>
</thead>
<tbody>
<tr>
<td>$g_1$</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>$g_2$</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>$g_3$</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>$g_4$</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>$g_5$</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>

Different scalings: different interpretations of the data

General problem

How to directly build a concept lattice from numerical data?
How to handle complex descriptions

An intersection as a similarity operator

- $\cap$ behaves as *similarity operator*

\[
\{m_1, m_2\} \cap \{m_1, m_3\} = \{m_1\}
\]

- $\cap$ induces an ordering relation $\subseteq$

\[
N \cap O = N \iff N \subseteq O
\]

\[
\{m_1\} \cap \{m_1, m_2\} = \{m_1\} \iff \{m_1\} \subseteq \{m_1, m_2\}
\]

- $\cap$ has the properties of a meet $\cap$ in a semi lattice, a commutative, associative and idempotent operation

\[
c \cap d = c \iff c \subseteq d
\]
Pattern structure

Given by \((G, (D, \sqcap), \delta)\)

- \(G\) a set of objects
- \((D, \sqcap)\) a semi-lattice of descriptions or patterns
- \(\delta\) a mapping such as \(\delta(g) \in D\) describes object \(g\)

A Galois connection

\[
A^\square = \bigcap_{g \in A} \delta(g) \quad \text{for } A \subseteq G
\]

\[
d^\square = \{g \in G \mid d \sqsubseteq \delta(g)\} \quad \text{for } d \in (D, \sqcap)
\]

S.O. Kuznetsov and A. Napoli

Tutorial on FCA at IJCAI 2013
Interval Pattern Structure

- A meet-semi-lattice for intervals \((D, \sqcap)\) where \(D\) is a set of intervals,
- a possible choice for the meet operator is the convexification of intervals:

\[
[a_1, b_1] \cap [a_2, b_2] = [\min(a_1, a_2), \max(b_1, b_2)]
\]

\[
[4, 5] \cap [5, 5] = [4, 5]
\]

\[
[a_1, b_1] \subseteq [a_2, b_2] \iff [a_2, b_2] \subseteq [a_1, b_1]
\]

\[
[4, 5] \subseteq [5, 5] \iff [5, 5] \subseteq [4, 5]
\]
Interval Pattern Structure

- An interval pattern $p$ is an $n$-dimensional vector of intervals: 
  $p = \langle [a_i, b_i] \rangle_{i \in [1,n]}$

- Operation $\cap$ and order of interval patterns:
  Given interval patterns $p = \langle [a_i, b_i] \rangle_{i \in [1,n]}$ and
  $q = \langle [c_i, d_i] \rangle_{i \in [1,n]}$:

  $p \cap q = \langle [a_i, b_i] \rangle_{i \in [1,n]} \cap \langle [c_i, d_i] \rangle_{i \in [1,n]}$
  $p \cap q = \langle [a_i, b_i] \cap [c_i, d_i] \rangle_{i \in [1,n]}$

- $p \cap q = p \iff p \subseteq q$
  $p \subseteq q \iff [a_i, b_i] \subseteq [c_i, d_i], \forall i \in [1, n]$
Interval pattern structures based on convexification

<table>
<thead>
<tr>
<th></th>
<th>$m_1$</th>
<th>$m_2$</th>
<th>$m_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$g_1$</td>
<td>5</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>$g_2$</td>
<td>6</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>$g_3$</td>
<td>4</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>$g_4$</td>
<td>4</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>$g_5$</td>
<td>5</td>
<td>8</td>
<td>5</td>
</tr>
</tbody>
</table>

\[
\{g_1, g_2\} = \bigcap_{g \in \{g_1, g_2\}} \delta(g) \\
= \langle 5, 7, 6 \rangle \sqcap \langle 6, 8, 4 \rangle \\
= \langle [5, 6], [7, 8], [4, 6] \rangle
\]

\[
\langle [5, 6], [7, 8], [4, 6] \rangle = \{g \in G | \langle [5, 6], [7, 8], [4, 6] \rangle \sqsubseteq \delta(g) \} \\
= \{g_1, g_2, g_5\}
\]

\[
(\{g_1, g_2, g_5\}, \langle [5, 6], [7, 8], [4, 6] \rangle ) \text{ is a pattern concept}
\]
Interval pattern concept lattice

- Highest concepts: largest extents and largest intervals (smallest intents)
- Lowest concepts: smallest extents and smallest intervals (largest intents)
- Problem: efficient pattern mining.
Links with conceptual scaling

Interordinal scaling [Ganter & Wille]

- A scale to encode intervals of attribute values
  
<table>
<thead>
<tr>
<th></th>
<th>$m_1 \leq 4$</th>
<th>$m_1 \leq 5$</th>
<th>$m_1 \leq 6$</th>
<th>$m_1 \geq 4$</th>
<th>$m_1 \geq 5$</th>
<th>$m_1 \geq 6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>5</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>6</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>

- Equivalent concept lattice

- Example
  
  \[
  (\{g_1, g_2, g_5\}, \{m_1 \leq 6, m_1 \geq 4, m_1 \geq 5, \ldots, \ldots\})
  \]
  
  \[
  (\{g_1, g_2, g_5\}, \langle [5, 6], \ldots, \ldots \rangle)
  \]

Why should we use pattern structures as we have scaling?
Processing a pattern structure is more efficient
Interval pattern search space

Counting all possible interval patterns with interordinal scaling

\[ \langle [a_{m_1}, b_{m_1}], [a_{m_2}, b_{m_2}], \ldots \rangle \]

where \( a_{m_i}, b_{m_i} \in W_{m_i} \)

\[ \prod_{i \in \{1, \ldots, |M|\}} \frac{|W_{m_i}| \times (|W_{m_i}| + 1)}{2} \]

360 possible interval patterns in our small example
Questions on interval pattern mining

- What are the links between numerical pattern structures and pattern mining?
- How can we reuse (good) ideas from pattern mining, i.e. closed patterns, generators and equivalence classes, in the framework of pattern structures?
Semantics for interval patterns

Interval patterns as (hyper) rectangles

<table>
<thead>
<tr>
<th></th>
<th>$m_1$</th>
<th>$m_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$g_1$</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>$g_2$</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>$g_3$</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>$g_4$</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>$g_5$</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>
Semantics for interval patterns

Interval patterns as (hyper) rectangles

<table>
<thead>
<tr>
<th></th>
<th>$m_1$</th>
<th>$m_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$g_1$</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>$g_2$</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>$g_3$</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>$g_4$</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>$g_5$</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

$\langle [4, 5], [5, 6] \rangle = \{ g_1, g_3, g_5 \}$
Semantics for interval patterns

Interval patterns as (hyper) rectangles

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$g_1$</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>$g_2$</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>$g_3$</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>$g_4$</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>$g_5$</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

$\delta(g_1) = {g_1, g_3, g_5}$

$\delta(g_2) = {g_1, g_3, g_5}$

$\delta(g_3) = {g_1, g_3, g_5}$

$\delta(g_4) = {g_1, g_3, g_5}$

$\delta(g_5) = {g_1, g_3, g_5}$
Semantics for interval patterns

Interval patterns as (hyper) rectangles

<table>
<thead>
<tr>
<th></th>
<th>$m_1$</th>
<th>$m_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$g_1$</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>$g_2$</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>$g_3$</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>$g_4$</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>$g_5$</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

$\langle [4, 5], [5, 6] \rangle = \{ g_1, g_3, g_5 \}$
$\langle [4, 5], [4, 7] \rangle = \{ g_1, g_3, g_5 \}$
$\langle [4, 5], [4, 6] \rangle = \{ g_1, g_3, g_5 \}$
Semantics for interval patterns

Interval patterns as (hyper) rectangles

<table>
<thead>
<tr>
<th></th>
<th>$m_1$</th>
<th>$m_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$g_1$</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>$g_2$</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>$g_3$</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>$g_4$</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>$g_5$</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

\[
\langle [4, 5], [5, 6] \rangle = \{ g_1, g_3, g_5 \} \\
\langle [4, 5], [4, 7] \rangle = \{ g_1, g_3, g_5 \} \\
\langle [4, 5], [4, 6] \rangle = \{ g_1, g_3, g_5 \} \\
\langle [4, 6], [5, 6] \rangle = \{ g_1, g_3, g_5 \}
\]
Semantics for interval patterns

Interval patterns as (hyper) rectangles

<table>
<thead>
<tr>
<th></th>
<th>$m_1$</th>
<th>$m_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$g_1$</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>$g_2$</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>$g_3$</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>$g_4$</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>$g_5$</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

$\langle [4, 5], [5, 6]\rangle = \{g_1, g_3, g_5\}$

$\langle [4, 5], [4, 7]\rangle = \{g_1, g_3, g_5\}$

$\langle [4, 5], [4, 6]\rangle = \{g_1, g_3, g_5\}$

$\langle [4, 6], [5, 6]\rangle = \{g_1, g_3, g_5\}$

$\langle [4, 5], [5, 7]\rangle = \{g_1, g_3, g_5\}$
Semantics for interval patterns

Interval patterns as (hyper) rectangles

$\langle [4, 5], [5, 6] \rangle \square = \{ g_1, g_3, g_5 \}$
$\langle [4, 5], [4, 7] \rangle \square = \{ g_1, g_3, g_5 \}$
$\langle [4, 5], [4, 6] \rangle \square = \{ g_1, g_3, g_5 \}$
$\langle [4, 6], [5, 6] \rangle \square = \{ g_1, g_3, g_5 \}$
$\langle [4, 5], [5, 7] \rangle \square = \{ g_1, g_3, g_5 \}$
$\langle [4, 6], [5, 7] \rangle \square = \{ g_1, g_3, g_5 \}$
A condensed representation

Equivalence classes of interval patterns

Two interval patterns with same image are said to be equivalent

\[ c \equiv d \iff c \Box = d \Box \]

Equivalence class of a pattern \( d \)

\[ [d] = \{ c | c \equiv d \} \]

- with a unique closed pattern: the smallest rectangle
- and one or several generators: the largest rectangles

In the example: 360 patterns; 18 closed patterns; 44 generators
**Algorithms: MintIntChange, MinIntChangeG[t|h]**

**Principle with an example**

1. Start from the most general interval pattern: \(<[4, 6], [7, 9], [4, 8]\rangle\)
2. Apply next minimal change following a canonical order \(c = \langle[4, 5], [7, 9], [4, 8]\rangle\)
3. Apply closure operator \(c^{\square\square} = \langle[4, 5], [7, 9], [5, 8]\rangle\)
4. If canonicity test fails: backtrack (in the depth first traversal)
5. Otherwise go to 2. with \(c^{\square\square} = \langle[4, 5], [7, 9], [5, 8]\rangle\)
Algorithms & experiments

Algorithms: MintIntChange, MinIntChangeG[t|h]

Experiments

- Mining several datasets from Bilkent University Repository
- Compression rate varies between $10^7$ and $10^9$
- Interordinal scaling
  - not efficient even with best algorithms (e.g. LCMv2)
  - redundancy problem discarding its use for generator extraction
Discussion

- Potential applications:
  - data privacy and $k$-anonymisation
  - $k$-box problem in computational geometry
  - quantitative association rule mining
  - data summarization

- Extension: focus on generator extraction

- Problems:
  - compression is not enough when considering very large data set
  - numerical data are noisy: this calls for fault-tolerant condensed representations
Introduction

A Smooth Introduction to Formal Concept Analysis
Derivation operators, formal concepts and concept lattice
The structure of the concept lattice
Scaling
Two algorithms for extracting the concepts and building the concept lattice

Relational Concept Analysis

Pattern Structures

Conclusion and References
Conclusion

- FCA is a well-founded mathematical theory equipped with efficient algorithmic tools.

- FCA is a polymorphic process and addresses problems ranging from knowledge discovery to knowledge representation and reasoning, and pattern recognition as well.

- FCA is rather mature and times are there for important variations, e.g. RCA and pattern structures (intervals and graphs).

- There is still room for many improvements, especially in dealing with trees and graphs, in taking into account domain knowledge, similarity, and in combining FCA with numerical processes.
Tools for building and visualizing concept lattices

- The Conexp program:
  http://sourceforge.net/projects/conexp

- The Galicia Platform:
  http://www.iro.umontreal.ca/~galicia/

- The Toscana platform:
  http://tockit.sourceforge.net/toscanaj/index.html

- The Formal Concept Analysis Homepage:
  http://www.upriss.org.uk/fca/fca.html
Elements of bibliography on concept lattices