Formal Concept Analysis

Part II

Radim BĚLOHLÁVEK

Dept. Computer Science
Palacky University, Olomouc
radim.belohlavek@acm.org

Applications of Formal Concept Analysis (FCA)

Applications of FCA – outline

- FCA as a method of data preprocessing,
- software for FCA,
- FCA in information retrieval,
- FCA in data analysis problems,
- links, resources.

FCA as a method of data preprocessing

- idea:
  input data $D \to$ (pre)processing of D by FCA $\to$ further processing (other methods),

  examples:
- FCA in factor analysis (formal concepts are optimal factors for Boolean factor analysis),
- FCA in mining association rules (enables mining non-redundant association rules),
- FCA in inductive logic programming (reducing the search space).

Formal Concepts and Their Role in Factor Analysis

What is factor analysis?

– Spearman: General intelligence, objectively determined and measured. Amer. J. Psychology (1904)

– according to Harman:
"The principal concern of factor analysis is the **resolution of a set of variables linearly in terms of (usually) a small number of categories or 'factors'**. . . . A satisfactory solution will yield factors which convey all the essential information of the original set of variables. Thus, the chief aim is to attain scientific parsimony or **economy of description**."

– given an objects $\times$ attributes $n \times m$ matrix I
– decompose I into $I \approx A \circ B$ where

- $A \ldots n \times k$ objects $\times$ factors matrix
- $B \ldots k \times m$ factors $\times$ attributes matrix
- desire: no. factors $<<$ no. attributes
- gain: objects described in space of k factors instead of m variables
- variables are manifestations of (more fundamental) factors

Formal Concepts and Their Role in Factor Analysis

example

input data (Rummel: Applied Factor Analysis, characteristics of hypothetical nations A–G, “p.c.”=“per capita”)

	GNP p.c. ($)	phones p.c.	vehicles p.c.	population (mil)	national income ($M)	area (mil km^2)
A	60	.004	.003	57.6	3,500	1.3
B	78	.004	.001	1.7	140	.04
C	85	.010	.008	2.3	198	.12
D	114	.083	.026	23.5	2,731	.97
E	321	.0122	.907	.8	303	.71
F	502	.679	.835	1.7	914	.63
G	1,361	1.421	.984	19.4	2,722	1.16

Can we find more general factors using which we could:

– describe the nations,

– explain all the variables (GNP, . . . , area)?

Denote by I the corresponding 7×6 matrix:

$$I = \begin{pmatrix} 60 & .004 & .003 & 57.6 & 3,500 & 1.3 \\ 78 & .004 & .001 & 1.7 & 140 & .04 \\ 85 & .010 & .008 & 2.3 & 198 & .12 \\ 114 & .083 & .026 & 23.5 & 2,731 & .97 \\ 321 & .0122 & .907 & .8 & 303 & .71 \\ 502 & .679 & .835 & 1.7 & 914 & .63 \\ 1,361 & 1.421 & .984 & 19.4 & 2,722 & 1.16 \end{pmatrix}$$

The question is: Can we decompose I into a product

$$I \approx A \circ B$$

where

- $\approx$ means "approximately equal",
- A is a $7 \times k$ matrix describing nations in terms of k factors (A_{il} …value of factor l on nation i), i.e., each nation is described by a k-dimensional vector of factors,
- B is a $k \times 6$ matrix describing factors in terms of original variables (B_{lj} …value of variable j on factor l), i.e., each factor is described by a 6-dimensional vector of original variables,
- $k < 6$ (number of factors < number of original variables).

Answer: yes, we can have $k = 2$

with $I \approx A \circ B$ being

$$\begin{pmatrix} 60 & .004 & .003 & 57.6 & 3,500 & 1.3 \\ 78 & .004 & .001 & 1.7 & 140 & .04 \\ 85 & .010 & .008 & 2.3 & 198 & .12 \\ 114 & .083 & .026 & 23.5 & 2,731 & .97 \\ 321 & .0122 & .907 & .8 & 303 & .71 \\ 502 & .679 & .835 & 1.7 & 914 & .63 \\ 1,361 & 1.421 & .984 & 19.4 & 2,722 & 1.16 \end{pmatrix} = \begin{pmatrix} -2.4 & 2.6 \\ -2.1 & -1.1 \\ -1.6 & -.4 \\ -.4 & 1.8 \\ .8 & -2.0 \\ 1.3 & -1.1 \\ 3.1 & 1.4 \end{pmatrix} \circ B$$

where B is a 2×6 matrix (we do not display B).

The two factors (columns of A) can be interpreted as:

- factor 1 . . . level of economic development
- factor 2 . . . size

Factor analysis (and related methods such as principal component analysis):

- classic topic,
- many textbooks available,
- implemented in SW packages.

Boolean Factor Analysis

Boolean factor analysis: data matrix I is a 0/1-matrix (Boolean matrix) of dimension $n \times m$, i.e. data consists of yes/no (presence/absence) variables such as

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

goal again: decompose $I \approx A \circ B$ where

- A ... objects $\times$ factors matrix, $n \times k$ matrix
- B ... factors $\times$ attributes matrix, $k \times m$ matrix
- desire: k (no. factors) $<<$ m (no. variables/attributes)

such as:

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \circ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Investigated since 1970s.

Factorizability and concept-factorizability

Definition (*k*-factorizability)

Boolean matrix I k-factorizable if there are Boolean matrices A $(n \times k)$ and B $(k \times m)$ s.t. $I = A \circ B$.

Example:

$$I = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

is 3-factorizable since

$$I = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \circ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Factorizability and concept-factorizability

Can we use (some) formal concepts $\langle \mathbf{A}, \mathbf{B} \rangle \in \mathcal{B}(\mathbf{X}, \mathbf{Y}, \mathbf{I})$ as factors?
(note: "factors = abstract concepts" appealing)

We will freely identify matrix I and the corresponding formal context, i.e. we consider $\langle X, Y, I \rangle$, $X = \{1, \dots, n\}$, $Y = \{1, \dots, m\}$, $\langle i, j \rangle \in I$ iff $I_{ij} = 1$.
Given matrix I and $\mathcal{F} = \{\langle A_1, B_1 \rangle, \dots, \langle A_k, B_k \rangle\} \subseteq \mathcal{B}(X, Y, I)$, denote by $A_{\mathcal{F}}$ and $B_{\mathcal{F}}$ the $n \times k$ and $k \times m$ Boolean matrices defined by

$$(A_{\mathcal{F}})_{il} = \begin{cases} 1 & \text{if } x_i \in A_l, \\ 0 & \text{if } x_i \notin A_l; \end{cases} \quad (B_{\mathcal{F}})_{lj} = \begin{cases} 1 & \text{if } y_j \in B_l, \\ 0 & \text{if } y_j \notin B_l. \end{cases}$$

Remark: A_i = i-th column of $A_{\mathcal{F}}$, B_i = i-th row of $B_{\mathcal{F}}$.

Definition (concept-factorizability, factor concepts)

Boolean matrix I concept-factorizable if there is $\mathcal{F} \subseteq \mathcal{B}(X, Y, I)$ s.t. $I = A_{\mathcal{F}} \circ B_{\mathcal{F}}$. Formal concepts from $\mathcal{F}$ are called factor concepts.

Example (concept-factorizability)

Take

$$I = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Consider formal concepts $\langle A_1, B_1\rangle = \langle\{x_1, x_2, x_3\}, \{y_1, y_2\}\rangle$, $\langle A_2, B_2\rangle = \langle\{x_3\}, \{y_1, y_2, y_3, y_4\}\rangle$, $\langle A_3, B_3\rangle = \langle\{x_2, x_4\}, \{y_1, y_5\}\rangle$. Denote

$$\mathcal{F} = \{\langle A_1, B_1\rangle, \langle A_2, B_2\rangle, \langle A_3, B_3\rangle\}.$$

Then

$$A_{\mathcal{F}} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad B_{\mathcal{F}} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Notice: extents of concepts from $\mathcal{F}$ are the columns of $A_{\mathcal{F}}$, intents are the rows of $B_{\mathcal{F}}$

Then

$$I = A_{\mathcal{F}} \circ B_{\mathcal{F}}.$$

Therefore, I is concept-factorizable with $\mathcal{F}$ being the set of concept-factors.

Optimality of concept-factorizability

Theorem (universality of concept-factorizability)

Each I is concept-factorizable. I.e., for each I there is $\mathcal{F}$ s.t. $I = A_{\mathcal{F}} \circ B_{\mathcal{F}}$.

Theorem (optimality of concept-factorizability)

If I is k-factorizable then I is concept-factorizable using $\mathcal{F}$ (factor concepts) s.t. $|\mathcal{F}| \leq k$.

Corollary (upper bound)

Each $n \times m$ Boolean matrix I is concept-factorizable using $\mathcal{F}$ with $|\mathcal{F}| \leq \min(n, m)$.

Proof of optimality theorem is based on

"Geometric interpretation" of formal concepts

Theorem (formal concepts = maximal rectangles)

$\langle A, B\rangle$ is a formal concept IFF $\langle A, B\rangle$ is a maximal rectangle in data.

I	y_1	y_2	y_3	y_4
x_1	1	1	1	1
x_2	1	0	1	1
x_3	0	1	1	1
x_4	0	1	1	1
x_5	1	0	0	0

I	y_1	y_2	y_3	y_4
x_1	1	1	1	1
x_2	1	0	1	1
x_3	0	1	1	1
x_4	0	1	1	1
x_5	1	0	0	0

I	y_1	y_2	y_3	y_4
x_1	1	1	1	1
x_2	1	0	1	1
x_3	0	1	1	1
x_4	0	1	1	1
x_5	1	0	0	0

$$(A_1, B_1) = (\{x_1, x_2, x_3, x_4\}, \{y_3, y_4\})$$

$$(A_2, B_2) = (\{x_1, x_3, x_4\}, \{y_2, y_3, y_4\})$$

$$(A_3, B_3) = (\{x_1, x_2\}, \{y_1, y_3, y_4\})$$

Further results on concept-factorizability

Attaining upper bounds of concept-factorizability

put

$$\begin{aligned}\mathcal{O}(X,Y,I) &= \{\langle \{x_i\}^{\uparrow\downarrow}, \{x_i\}^{\uparrow}\rangle \mid 1 \le i \le n\} \subseteq \mathcal{B}(X,Y,I),\\ \mathcal{A}(X,Y,I) &= \{\langle \{y_j\}^{\downarrow}, \{y_j\}^{\downarrow\uparrow}\rangle \mid 1 \le j \le m\} \subseteq \mathcal{B}(X,Y,I).\end{aligned}$$

Theorem (particular $\mathcal{F}$ which is not worse than upper bound)

Let $\mathcal{F} = \mathcal{O}(\mathcal{X},\mathcal{Y},\mathcal{I})$ or $\mathcal{F} = \mathcal{A}(\mathcal{X},\mathcal{Y},\mathcal{I})$, whichever is smaller. Then $|\mathcal{F}| \le \min(n,m)$ and I is concept-factorizable using $\mathcal{F}$.

Mandatory factor-concepts

Theorem (concepts from $\mathcal{O}(X,Y,I) \cap \mathcal{A}(X,Y,I)$ are always factor concepts, no choice)

Let I be concept-factorizable with a set $\mathcal{F}$ of factor concepts. Then $\mathcal{O}(X,Y,I) \cap \mathcal{A}(X,Y,I) \subseteq \mathcal{F}$.

Algorithm for computing factor concepts

previous results $\Longrightarrow$ algorithm for computing a minimal set of factor concepts

INPUT: Boolean matrix I
OUTPUT: set $\mathcal{F}$ of factor concepts (desire: $\mathcal{F}$ is small)

basic points:

- compute concept lattice $\mathcal{B}(X, Y, I)$
  (algorithm with polynomial time delay exists)
- finding factor concepts can be **reduced to set-covering problem**
  (approximation algorithms exist)
- theoretical insight (e.g. mandatory factors) speeds-up set-covering algorithms

Illustrative example: input data

I ... 12×8 Boolean matrix describing patients $\times$ symptoms

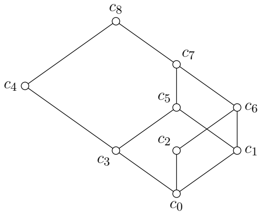
$$I = \begin{pmatrix} 1&1&1&0&1&0&0&0\\ 1&1&0&0&0&1&0&1\\ 0&1&0&0&1&0&1&0\\ 1&1&0&0&0&1&0&1\\ 1&1&1&0&1&0&0&0\\ 0&1&0&0&1&0&1&0\\ 0&1&0&0&1&0&1&0\\ 0&0&0&0&0&0&1&0\\ 1&1&1&0&1&0&0&0\\ 0&0&0&0&0&0&1&0\\ 1&1&1&0&1&0&0&0\\ 1&1&0&0&0&1&0&1 \end{pmatrix}$$

symptom	symptom description
y_1	headache
y_2	fever
y_3	painful limbs
y_4	swollen glands in neck
y_5	cold
y_6	stiff neck
y_7	rash
y_8	vomiting

Illustrative example: corresponding concept lattice

recall: concept lattice of I = space of possible factors

diagram of concept lattice:



formal concepts (possible factors):

c_i	⟨set of patients⟩,⟨set of symptoms ⟩	verbal description
c_0	$\langle\{\},\{y_1,y_2,y_3,y_4,y_5,y_6,y_7,y_8\}\rangle$	empty concept
c_1	$\langle\{x_1,x_5,x_9,x_{11}\},\{y_1,y_2,y_3,y_5\}\rangle$	"flu"
c_2	$\langle\{x_2,x_4,x_{12}\},\{y_1,y_2,y_6,y_8\}\rangle$	"meningitis"
c_3	$\langle\{x_3,x_6,x_7\},\{y_2,y_5,y_7\}\rangle$	"measles"
c_4	$\langle\{x_3,x_6,x_7,x_8,x_{10}\},\{y_7\}\rangle$	"chickenpox"
c_5	$\langle\{x_1,x_3,x_5,x_6,x_7,x_9,x_{11}\},\{y_2,y_5\}\rangle$	"suspicion of flu or measles"
c_6	$\langle\{x_1,x_2,x_4,x_5,x_9,x_{11},x_{12}\},\{y_1,y_2\}\rangle$	"suspicion of flu or meningitis"
c_7	$\langle\{x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_9,x_{11},x_{12}\},\{y_2\}\rangle$	"susp. of flu or meas. or men."
c_8	$\langle\{x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9,x_{10},x_{11},x_{12}\},\{\}\rangle$	universal concept

Illustrative example: input data factorized

two minimal sets of factor concepts are:

$$\mathcal{F} = \{c_1, c_2, c_3, c_4\} \quad \text{and} \quad \mathcal{F}' = \{c_1, c_2, c_4, c_5\}.$$

Thus, I can be decomposed by

$$I = A_{\mathcal{F}} \circ B_{\mathcal{F}} \quad \text{or} \quad I = A_{\mathcal{F}'} \circ B_{\mathcal{F}'}.$$

For $I = A_{\mathcal{F}} \circ B_{\mathcal{F}}$, we have

$$\begin{pmatrix} 1&1&1&0&1&0&0&0\\ 1&1&0&0&0&1&0&1\\ 0&1&0&0&1&0&1&0\\ 1&1&0&0&0&1&0&1\\ 1&1&1&0&1&0&0&0\\ 0&1&0&0&1&0&1&0\\ 0&1&0&0&1&0&1&0\\ 0&0&0&0&0&0&1&0\\ 1&1&1&0&1&0&0&0\\ 0&0&0&0&0&0&1&0\\ 1&1&1&0&1&0&0&0\\ 1&1&0&0&0&1&0&1 \end{pmatrix} = \begin{pmatrix} 1&0&0&0\\ 0&1&0&0\\ 0&0&1&1\\ 0&1&0&0\\ 1&0&0&0\\ 0&0&1&1\\ 0&0&1&1\\ 0&0&0&1\\ 1&0&0&0\\ 0&0&0&1\\ 1&0&0&0\\ 0&1&0&0 \end{pmatrix} \circ \begin{pmatrix} 1&1&1&0&1&0&0&0\\ 1&1&0&0&0&1&0&1\\ 0&1&0&0&1&0&1&0\\ 0&0&0&0&0&0&1&0 \end{pmatrix}.$$

Conclusions and further topics

- FCA brings
  - foundations for Boolean factor analysis
  - factors = formal concepts (psychological plausibility, easy interpretation)
  - optimality results and theoretical insights for algorithms
- further issues and problems:
  - heuristics for finding sets of factor concepts,
  - approximate factorizability, i.e. $I \approx A \circ B$, instead of $I = A \circ B$
  - factor analysis of matrices with truth degrees like

    $$\begin{pmatrix} 0.1 & 1 & 0.8 & 0.8 \\ 1 & 1 & 0.2 & 0.5 \\ 1 & 0.7 & 0.5 & 1 \end{pmatrix}$$

    - analogous results using fuzzy concept lattices
    - alternative to classical factor analysis (nonlinear)

Concept-factorizability – revisited with proofs

Note: $X = \{1, \dots, n\}$, $Y = \{1, \dots, m\}$, I denotes both the matrix and the relation between X and Y.

Theorem (universality of concept-factorizability)

Each I is concept-factorizable. I.e., for each I there is $\mathcal{F}$ s.t. $I = A_\mathcal{F} \circ B_\mathcal{F}$.

Proof.

Very easy proof is the following:
Take $\mathcal{F} = \mathcal{B}(X, Y, I)$ (all formal concepts). Such $\mathcal{F}$ is usually not optimal (might be very large) but it serves the proof. Denote $k = |\mathcal{B}(X, Y, I)|$.
We need to show $A_\mathcal{F} \circ B_\mathcal{F} = I$, i.e., need to show

$$(A_\mathcal{F} \circ B_\mathcal{F})_{ij} = 1 \text{ iff } I_{ij} = 1. \text{ We have:}$$

$(A_\mathcal{F} \circ B_\mathcal{F})_{ij} = 1$ iff
$\max_{l=1}^{k} \min((A_\mathcal{F})_{il}, (B_\mathcal{F})_{lj}) = 1$ iff
exists l s.t. $(A_\mathcal{F})_{il} = 1$ and $(B_\mathcal{F})_{lj} = 1$ iff
exists $\langle A_l, B_l \rangle \in \mathcal{B}(X, Y, I)$ s.t. $i \in A_l$ and $j \in B_l$ iff
$I_{ij} = 1$. □

Concept-factorizability – revisited with proofs

Proof of optimality theorem gives us insight about what it means to find a set $\mathcal{F}$ of factor concepts. The proof is a "proof by pictures"-type.

Theorem (optimality of concept-factorizability)

If I is k-factorizable then I is concept-factorizable using $\mathcal{F}$ (factor concepts) s.t. $|\mathcal{F}| \leq k$.

First, consider the meaning of $I = A \circ B$. By definition,

$$I_{ij} = \max_{l=1}^{k} \min(A_{il}, B_{lj}),$$

i.e.

$$I_{ij} = \min(A_{i1}, B_{1j}) \text{ OR } \cdots \text{ OR } \min(A_{ik}, B_{kj}),$$

which can be rewritten as

$$I = A_{\_1} \circ B_{1\_} \text{ OR } \cdots \text{ OR } A_{\_k} \circ B_{k\_}$$

where

–$A_{\_l}$ is the l-th column of A,

–$B_{l\_}$ is the l-th row of B.

Concept-factorizability – revisited with proofs

Example: $I = A \circ B$ written as $I = A_{\_1} \circ B_{1\_}$ OR $\cdots$ OR $A_{\_k} \circ B_{k\_}$

$$\begin{pmatrix} 1&1&0&0&0\\ 1&1&0&0&1\\ 1&1&1&1&0\\ 1&0&0&0&1 \end{pmatrix} = \begin{pmatrix} 1&0&0&1\\ 1&0&1&0\\ 1&1&0&0\\ 0&0&1&0 \end{pmatrix} \circ \begin{pmatrix} 1&1&0&0&0\\ 0&0&1&1&0\\ 1&0&0&0&1\\ 0&1&0&0&0 \end{pmatrix}.$$

can be written as

$$\begin{pmatrix} 1&1&0&0&0\\ 1&1&0&0&1\\ 1&1&1&1&0\\ 1&0&0&0&1 \end{pmatrix} = \begin{pmatrix} 1\\1\\1\\0 \end{pmatrix} \circ \begin{pmatrix} 1&1&0&0&0 \end{pmatrix} \text{ OR } \begin{pmatrix} 0\\0\\1\\0 \end{pmatrix} \circ \begin{pmatrix} 0&0&1&1&0 \end{pmatrix} \text{ OR}$$

$$\text{OR } \begin{pmatrix} 0\\1\\0\\1 \end{pmatrix} \circ \begin{pmatrix} 1&0&0&0&1 \end{pmatrix} \text{ OR } \begin{pmatrix} 1\\0\\0\\0 \end{pmatrix} \circ \begin{pmatrix} 0&1&0&0&0 \end{pmatrix}, \text{ which gives}$$

$$\begin{pmatrix} 1&1&0&0&0\\ 1&1&0&0&1\\ 1&1&1&1&0\\ 1&0&0&0&1 \end{pmatrix} = \begin{pmatrix} 1&1&0&0&0\\ 1&1&0&0&0\\ 1&1&0&0&0\\ 0&0&0&0&0 \end{pmatrix} \text{OR} \begin{pmatrix} 0&0&0&0&0\\ 0&0&0&0&0\\ 0&0&1&1&0\\ 0&0&0&0&0 \end{pmatrix} \text{OR} \begin{pmatrix} 0&0&0&0&0\\ 1&0&0&0&1\\ 0&0&0&0&0\\ 1&0&0&0&1 \end{pmatrix} \text{OR} \begin{pmatrix} 0&1&0&0&0\\ 0&0&0&0&0\\ 0&0&0&0&0\\ 0&0&0&0&0 \end{pmatrix}.$$

Concept-factorizability – revisited with proofs

now look at

$$\begin{pmatrix} 1&1&0&0&0\\ 1&1&0&0&1\\ 1&1&1&1&0\\ 1&0&0&0&1 \end{pmatrix} = \begin{pmatrix} 1&1&0&0&0\\ 1&1&0&0&0\\ 1&1&0&0&0\\ 0&0&0&0&0 \end{pmatrix} \text{OR} \begin{pmatrix} 0&0&0&0&0\\ 0&0&0&0&0\\ 0&0&1&1&0\\ 0&0&0&0&0 \end{pmatrix} \text{OR} \begin{pmatrix} 0&0&0&0&0\\ 1&0&0&0&1\\ 0&0&0&0&0\\ 1&0&0&0&1 \end{pmatrix} \text{OR} \begin{pmatrix} 0&1&0&0&0\\ 0&0&0&0&0\\ 0&0&0&0&0\\ 0&0&0&0&0 \end{pmatrix}.$$

All the matrices connected by OR correspond to rectangles filled with 1's.

Therefore:
$I = A \circ B$ with k being the inner dimension (as above) means that $I =$ OR-composition of k rectangles filled with 1's!
Each rectangle can be represented by an $n \times 1$ column $A_{\_l}$ of A, and a $1 \times m$ row $B_{l\_}$ of B.

Now, proof of optimality theorem is easy. Let $I = A \circ B$. Since formal concepts of $\mathcal{B}(X, Y, I)$ are just the maximal rectangles contained in I, each rectangle represented by column $A_{\_l}$ of A and row $B_{l\_}$ of B is contained in some maximal rectangle, i.e., in some concept $\langle C, D\rangle \in \mathcal{B}(X, Y, I)$ (in that $A_{\_l} \subseteq C$ and $B_{l\_} \subseteq D$).
Denote by $\mathcal{F}$ the set of all formal concepts $\langle C, D\rangle \in \mathcal{B}(X, Y, I)$ which we need for all rectangles $A_{\_l}$, $B_{l\_}$ ($l = 1 \dots k$). Obviously, we need at most k formal concepts but may need less than k (since two different rectangles may be covered by a single formal concept). This gives:

$$|\mathcal{F}| \leq k.$$

□

Concept-factorizability – revisited with proofs

From the insight given by the proof of optimality theorem:
Finding a set $\mathcal{F}$ of factor concepts = finding a set of maximal rectangles in I which cover I. As an example:

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Possible solution: $\langle A_1, B_1 \rangle = \langle \{x_1, x_2, x_3\}, \{y_1, y_2\} \rangle$,
$\langle A_2, B_2 \rangle = \langle \{x_3\}, \{y_1, y_2, y_3, y_4\} \rangle$, $\langle A_3, B_3 \rangle = \langle \{x_2, x_4\}, \{y_1, y_5\} \rangle$
correspond to maximal rectangles

$$\begin{pmatrix} \mathbf{1} & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & \mathbf{1} & 0 & 0 & 1 \\ \mathbf{1} & \mathbf{1} & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad \text{and} \quad \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ \mathbf{1} & 1 & 0 & 0 & \mathbf{1} \\ 1 & 1 & 1 & 1 & 0 \\ \mathbf{1} & 0 & 0 & 0 & \mathbf{1} \end{pmatrix}.$$

$\Rightarrow$ Looking for a minimal set of factor concepts is a particular instance of set-covering problem.
Algorithms exist for solving set-covering problem!

Concept-factorizability – set-covering problem

set-covering problem:

INPUT: set U, subset $V \subseteq U$, collection $P \subseteq 2^U$.
OUTPUT: minimal (w.r.t. number of its elements) covering $C \subseteq P$ of V (i.e., we require $\bigcup_{Q \in C} Q = V$).

Example

$U = \{1, 2, \ldots, 10\}$, $V = \{2, 4, 6, 8, 10\}$, $P = \{\{1, 2\}, \{2, 3\}, \{4, 5\}, \{6, 7, 8\}, \{9, 10\}, \{1, 3, 5\}, \{2, 4\}, \{4, 6\}, \{8, 9, 10\}\}$.

- $C = \{\{1, 2\}, \{8, 9, 10\}\}$ is not a covering of V because $\bigcup C \neq V$.
- $C = \{\{1, 2\}, \{2, 3\}, \{4, 5\}, \{6, 7, 8\}, \{9, 10\}\}$ is a covering of V because $\bigcup C = V$. But C is not minimal because there exist coverings of V which contain smaller number of sets.
- $C = \{\{2, 4\}, \{6, 7, 8\}, \{8, 9, 10\}\}$ is a minimal covering of V because $\bigcup C = V$ and no other covering has a smaller number of sets than 3.
- $C = \{\{2, 4\}, \{4, 6\}, \{8, 9, 10\}\}$ is another minimal covering of V.

Concept-factorizability – set-covering problem

reducing the problem of finding a minimal set of factor-concepts to set-covering problem:

Theorem

Given $\langle X, Y, I\rangle$ (input table, input binary matrix), $\mathcal{F} \subseteq \mathcal{B}(X, Y, I)$ is a minimal set of factor-concepts iff $\mathcal{F}$ is a solution to a minimal set-covering problem where:
$U = X \times Y$, $V = I$, $P = \{A \times B \mid \langle A, B\rangle \in \mathcal{B}(X, Y, I)\}$.

Proof.

Immediately from previous considerations. $\square$

Concept-factorizability – set-covering problem

Example (translating search for factor-concepts to set-covering problem)

For

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix},$$

all formal concepts are: $\langle A_1, B_1 \rangle = \langle \{x_1, x_2, x_3\}, \{y_1, y_2\} \rangle$, $\langle A_2, B_2 \rangle = \langle \{x_3\}, \{y_1, y_2, y_3, y_4\} \rangle$, $\langle A_3, B_3 \rangle = \langle \{x_2, x_4\}, \{y_1, y_5\} \rangle$, $\langle A_4, B_4 \rangle = \langle \{x_2\}, \{y_1, y_2, y_5\} \rangle$, $\langle A_5, B_5 \rangle = \langle \{x_1, x_2, x_3, x_4\}, \{y_1\} \rangle$, $\langle A_6, B_6 \rangle = \langle \emptyset, Y \rangle$.
$I = \{\langle x_1, y_1 \rangle, \langle x_1, y_2 \rangle, \langle x_2, y_1 \rangle, \ldots, \langle x_4, y_5 \rangle\}$, $P = \{A_1 \times B_1, \ldots, A_6 \times B_6\}$, and

- $A_1 \times B_1 = \{\langle x_1, y_1 \rangle, \langle x_1, y_2 \rangle, \langle x_2, y_1 \rangle, \langle x_2, y_2 \rangle, \langle x_3, y_1 \rangle, \langle x_3, y_2 \rangle\}$.
- $\ldots$
- $A_6 \times B_6 = \emptyset$.

Applications of FCA and software for FCA

- useful links can be found at
  `http://www.upriss.org.uk/fca/fca.html` ("FCA Homepage")
  - bibliography
  - conferences (past and upcoming)
  - mailing list
  - software
  - websites, websites of related disciplines
  - introductory material
- Wikipedia link
  `http://en.wikipedia.org/wiki/Formal_concept_analysis`
- papers available on the web

Software for FCA

- software for computing concept lattices,
- software for computing attribute implications,
- software for drawing concept lattices,
- interface to databases and other software,
- links from `http://www.upriss.org.uk/fca/fca.html`,
- ToscanaJ
  - best developed, at sourceforge: `http://tockit.sourceforge.net/`
  - part of software for Conceptual Knowledge Processing,
  - consists of
    - ToscanaJ ("viewer/browser component"),
    - Elba ("editor for conceptual schemas on relational databases"),
    - Lucca ("experimental editor, makes use of implication analysis of SQL clauses to allow very explorative and intuitive creation of database-connected systems").
    - can be downloaded from `http://toscanaj.sourceforge.net/downloads/index.html`

FCA in Information Retrieval

pioneering work of R. Godin; C. Carpineto, G. Romano

detailed treatment in Carpineto C., Romano G.: Concept Data Analysis. Wiley, 2004 (Chap. 3, 4).

Information Retrieval (IR) = iterative and interactive process, retrieval of required information from data (example: search by keywords, retrieval of documents):

- submitting query,
- looking at the documents returned,
- submitting a refined query until appropriate documents are found.

rationale behind using FCA in IR:

- current search engines (Google, Yahoo, etc.) provide a ranked list of retrieved documents (provide “simplistic” linear view on retrieved information),
- FCA enables structured view of retrieved information,
- user is supplied with a (part of a) concept lattice of retrieved documents.

FCA in Information Retrieval

basic ideas (taken from CREDO architecture):

- submitting query,by a user,
- transforming query to to a format (such as SOAP) which can be sent to a Web search engine (Google, Yahoo),
- submitting query to Web search engine, receiving results (typically in XML format),
- parsing results and indexing the document terms,
- establishing formal context (objects=documents, attributes=index terms),
- computing concept lattice and displaying it to the user

FCA in Information Retrieval

CREDO

- system for Conceptual REorganization of DOcuments (developed by Carpineto and Romano at Fondazione Ugo Bordoni, Italy)
- Carpineto C., Romano G.: Exploiting the Potential of Concept Lattices for Information Retrieval with CREDO. J. Universal Computer Science 10(2004), 985–1013 `http://www.fub.it/repository/riviste/JUCS04.pdf`
- Search tool available at `http://credo.fub.it`.
- CREDINO (mobile version, `http://credino.dimi.uniud.it/`),

Illustration:

- search for "jaguar" (Carpineto and Romano's example, ambiguous term), Credo vs. Yahoo or Google,
- search for "xml",
- search for "formal concept analysis",
- search for "radim belohlavek"

FCA in Information Retrieval

FooCA

- developed by Bjoern Koester at (Webstrategy GmbH, Darmstadt; TU Dresden, Germany), `http://www.bjoern-koester.de/`
- B. Koester: FooCA - Web Information Retrieval with Formal Concept Analysis. Verlag Allgemeine Wissenschaft, Mhltal, 2006, ISBN 9783-935924-06-1.
- presents search results directly in a form of labeled Hasse diagram (clicking on the nodes opens a browser window with URLs),
- `http://fooca.webstrategy.de/` - requires username and password,
- overview in: B. Koester: Conceptual Knowledge Retrieval with FooCA: Improving Web Search Engine Results with Contexts and Concept Hierarchies at `http://www.bjoern-koester.de/bjoern_koester_conceptual_knowledge_retrieval_springer_icdm_2006.pdf`

FCA in Software Engineering and Web Ontologies

- various software engineering constructions resemble concept hierarchies,
- examples: object-oriented design (class hierarchy=hierarchy of concepts), hierarchical organization of software modules, etc.,
- rationale behind using FCA in SWEng: hierarchical constructions = (parts of) concept lattices.

sample papers:

- G. Snelting, F. Tip: Understanding Class Hierarchies Using Concept Analysis. ACM Transactions on Programming Languages and Systems, May 2000, pp. 540-582. (available in ACM digital library)
  - analyzing and re-engineering class hierarchies,
  - objects=program variables used to access classes, attributes=class members,
  - resulting concept lattice shows how class members are used and suggests a new (non-redundant, more efficient) class hierarchy; concepts intents are groups of class members which "belong together" (are accessed used by common variables),

quote from G. Snelting, F. Tip: Understanding Class Hierarchies Using Concept Analysis. ACM Transactions on Programming Languages and Systems, May 2000, pp. 540-582:
“In our approach, a class hierarchy is processed along with a set of applications that use it, and a fine-grained analysis of the access and subtype relationships between objects, variables, and class members is performed. The result of this analysis is again a class hierarchy, which is guaranteed to be behaviorally equivalent to the original hierarchy, but in which each object only contains the members that are required. Our method is semantically well-founded in concept analysis: the new class hierarchy is a minimal and maximally factorized concept lattice that reflects the access and subtype relationships between variables, objects and class members. The method is primarily intended as a tool for finding imperfections in the design of class hierarchies, and can be used as the basis for tools that largely automate the process of reengineering such hierarchies. The method can also be used as a space-optimizing source-to-source transformation that removes redundant fields from objects. A prototype implementation for Java has been constructed, and used to conduct several case studies.”

FCA in Software Engineering and Web Ontologies

- Tonella, P.: Formal concept analysis in software engineering. Proc. ICSE 2004. (available in IEEE Explore)
  - survey,
  - quote: "Given a binary relationship between objects and attributes, concept analysis is a powerful technique to organize pairs of related sets of objects and attributes into a concept lattice, where higher level concepts represent general features shared by many objects, while lower level concepts represent the object-specific features. Concept analysis was recently applied to several software engineering problems, such as: restructuring the code into more cohesive components, identifying class candidates, locating features in the code by means of dynamic analysis, reengineering class hierarchies. This paper provides the background knowledge required by such applications. Moreover, the methodological issues involved in the different applications of this technique are considered by giving a detailed presentation of three of them: module restructuring, design pattern inference and impact analysis based on decomposition slicing. The paper is concluded by an overview on other kinds of applications."

FCA in Software Engineering and Web Ontologies

– Snelting, G.: Concept analysisa new framework for program understanding Proc. 1998 ACM SIGPLAN-SIGSOFT, pp. 1–10. (available in ACM digital library).
– Pfaltz J. L.: Using Concept Lattices to Uncover Causal Dependencies in Software. Proc. ICFCA 2006, Springer, pp. 233–247 (avaiable at `http://www.cs.virginia.edu/~jlp/06.FCA.pdf`).
– Lindig C., Snelting G.: Assessing modular structure of legacy code based on mathematical concept analysis. Proc. of the 19th international conference on Software engineering, Boston, Massachusetts, pp. 349–359, 1997 (`http://www.st.cs.uni-sb.de/publications/files/lindig-icse-1997.pdf`).
– Vinod Ganapathy, David King and Trent Jaeger, Somesh Jha. Mining Security-Sensitive Operations in Legacy Code using Concept Analysis. Proc. 29th Int. Conf. on Software Engineering, Minneapolis, Minnesota, May 2007, (`http://www.cs.wisc.edu/~vg/papers/icse2007/icse2007.pdf`).

FCA in Homeland Security

- quite recent topic,
- New York Times 2006 and San Francisco Chronicle 2006 papers (`http://www.sfgate.com/cgi-bin/article.cgi?file=/chronicle/archive/2006/07/09/INGIVJQ75N1.DTL`)
- started in Los Alamos National Lab,
- Voss, Susan and Cliff Joslyn: Advanced Knowledge Integration in Assessing Terrorist Threats, LANL Technical Report LAUR 02-7867.
- Joslyn, Cliff and Mniszewski, Susan: Relational Analytical Tools: DataDelver and Formal Concept Analysis, LANL Technical Report 02-7697. (`ftp://ftp.c3.lanl.gov/pub/users/joslyn/hl1.pdf`).
- Conference: Mathematical Methods in Counterterrorism (2005, 2006).