Formal Concept Analysis

III Knowledge Discovery

Robert Jäschke
Asmelash Teka Hadgu

FG Wissensbasierte Systeme/L3S Research Center
Leibniz Universität Hannover

slides based on a lecture by Prof. Gerd Stumme
Background Knowledge

- Simplifying Implications of the Stem Base
- Optimizing the Computation of the Stem Base
- Incomplete Knowledge About Objects
- “Harmless” Background Knowledge
- “Difficult” Background Knowledge
First of all: the stem base is non-redundant, i.e., we *can not remove implications*

But: *redundancy in the premise or the conclusion* can be removed

*One* redundant attribute in the premise or the conclusion we can just remove:

- Since $a \rightarrow b, c$ we can simplify $d, e \rightarrow a, b, c$ to $d, e \rightarrow a, b$

*Several* redundant attributes can not always be removed:

- $a$ and $c$ can not be removed from $d, e \rightarrow a, b, c$

---

**exemplary stem base**

\[
\begin{align*}
  a & \rightarrow b, c \\
  d, e & \rightarrow a, b, c \\
  c, e & \rightarrow a, b, d \\
  c, d & \rightarrow a, b, e 
\end{align*}
\]
Assume that we have computed the implication \( \{c, d\} \rightarrow \{a, b, e\} \) (of the attribute set \( \{a, b, c, d, e\} \)):

- then \textsc{Next Closure} checks \( \{c, d\} <_e \{a, b, c, d, e\} \) – which fails
- \textit{improvement}: directly continue with \( i := b \)

Similar, after the implication \( \{a\} \rightarrow \{b, c\} \):

- \textsc{Next Closure} is unsuccessfully checking \( \{a\} <_e \{a, b, c, e\} \), \( \{a\} <_d \{a, b, c, d\} \), \( \{a\} <_c \{a, b, c\} \)
- \textit{improvement}: directly continue with \( A := \{a, b, c\} \)
In general: Let \( k := \max \mathcal{L}^\bullet(A + i) \) and \( l := \min(\mathcal{L}^\bullet(A + i))'' \setminus \mathcal{L}^\bullet(A + i) \)

- \( l < k \): ignore all \( i > k \) and continue with \( i < k \)
  - In the example \( \{c, d\} \rightarrow \{a, b, e\} : \)
    \[ d = \max\{c, d\}, a = \min\{a, b, c, d, e\} \setminus \{c, d\} \text{ – ignore } e \text{ and } d \]
  
- \( k < l \): continue directly with \( A := A'' \)
  - Proposition: If \( P \) is a pseudo-intent and no element of \( P'' \setminus P \) is smaller than any element of \( P \), then \( P \) is the lexically largest pseudo-intent with the closure \( P'' \).
  - In the example \( \{a\} \rightarrow \{b, c\} \) we can continue with \( \{a\}'' = \{a, b, c\} \), instead of \( \{a\} \) (otherwise we would unsuccessfully try \( i := e, d, c \))
Incomplete Knowledge About Objects

Previously:

![Image of a computer window with a question and options]

**Question:** Is it true, that when object has attribute(s) >10Mill Einwohner, that it also has attribute(s) EU, NATO?

**Options:**
- Yes
- No
- Stop Attribute Exploration

**Response:** No, the Ukraine is not a member of the EU!

**Additional Thought:** But is it part of the Schengen area?
Incomplete Knowledge About Objects

- First, we start with a context $E = (E, M, J)$ with examples $E$ of a context $(G, M, I)$ (i.e., $E \subseteq G$ and $J := I \cap (E \times M)$)
- Then, we replace $E$ by $E_+ = (E, M, J_+)$ and $E_? = (E, M, J_?)$ with $J_+ \subseteq J \subseteq J_?$
- $(E_+, E_?)$ is called *partial formal context*
- For each example object $e \in E$ we have then three sets of attributes:
  - $e^+$ – the attributes $e$ is *known to have*
  - $e^? \supseteq g^+$ – the attributes $e$ *may have*
  - $e^- := M \setminus g^?$ – the attributes $e$ is *known not to have*
Incomplete Knowledge About Objects

- Instead of a complete example $e$, it is sufficient to supply $e^+$ and $e^-$
- $e^+ \text{ und } e^? := M \setminus e^-$ are added to $E_+$ and $E_?$, respectively
- For $B := \mathcal{L}^\bullet(A + i)$ we compute instead of $B^{''}$:
  \[
  B^+? := \bigcap \{e^? | e \in E, B \subseteq e^+ \}
  \]
  (that’s not a closure operator!)
- Any modification of the list $\mathcal{L}$ leads to a modification of $E_+$ and $E_?$:
  - for each $e \in E$ we replace $e^+$ by $\mathcal{L}(e^+)$
  - for each $e \in E$ we successively remove those elements $m$ which do not satisfy the condition $\mathcal{L}(e^+ \cup \{m\}) \subseteq e^?$
Upon completion of the algorithm we have that

- \( \mathcal{L} \) is the stem base of \((G, M, I)\)
- \( e^{++} = e^{II} \)

On the blackboard: Countries of Europe

- as attributes this time only EU, €, Schengen, NATO
- Let's start with Germany . . .
Background Knowledge

A not so nice example:

Possible outcomes of a driving test

<table>
<thead>
<tr>
<th></th>
<th>theory</th>
<th>driving</th>
<th>license</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>pass</td>
<td>fail</td>
<td>pass</td>
</tr>
<tr>
<td>1</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>2</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>3</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>4</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>

The stem base for the context

- driving = fail → license = fail
- theory = fail → license = fail
- license = fail, driving = pass → theory = fail
- license = fail, theory = pass → driving = fail
- driving = pass, theory = pass → license = pass
- license = pass → driving = pass, theory = pass
- license = fail, theory = fail, driving = pass, driving = fail → ⊥
- license = fail, theory = fail, theory = pass, driving = fail → ⊥

Wouldn’t we rather expect

theory = pass, driving = pass ↔ license = pass?
This does not work, because we intuitively assume that “fail” is the negation of “pass”, i.e., we assume that

\[ \text{pass, fail} \rightarrow \bot \quad \text{and} \quad \top \rightarrow \text{pass or fail} \]

hold as \textit{background knowledge} for all parts of the test.

- \( \text{pass, fail} \rightarrow \bot \) is an \textit{implication}
- \( \top \rightarrow \text{pass or fail} \) is a \textit{clause}
Can we add (during attribute exploration) further (correct) objects and implications?

- Yes, we can add objects at any time!
- We can also add implications.
- **But:** the computed implications are then not the stem base of the context \((G, M, I)\)!
- Instead, we get a base \textit{relative} to the manually added implications.
- If we add implications during exploration, the resulting set of implications is not necessarily redundant.
If we include \( \text{pass, fail} \rightarrow \bot \) (i.e., the three implications “theory = pass, theory = fail \rightarrow \bot”, etc.) as background knowledge, we get a base with six implications:

<table>
<thead>
<tr>
<th></th>
<th>theory</th>
<th>driving</th>
<th>license</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>pass</td>
<td>fail</td>
<td>pass</td>
</tr>
<tr>
<td>1</td>
<td>( \times )</td>
<td>( \times )</td>
<td>( \times )</td>
</tr>
<tr>
<td>2</td>
<td>( \times )</td>
<td>( \times )</td>
<td>( \times )</td>
</tr>
<tr>
<td>3</td>
<td>( \times )</td>
<td>( \times )</td>
<td>( \times )</td>
</tr>
<tr>
<td>4</td>
<td>( \times )</td>
<td>( \times )</td>
<td>( \times )</td>
</tr>
</tbody>
</table>

- \( \text{driving = fail} \rightarrow \text{license = fail} \)
- \( \text{theory = fail} \rightarrow \text{license = fail} \)
- \( \text{license = fail, driving = pass} \rightarrow \text{theory = fail} \)
- \( \text{license = fail, theory = pass} \rightarrow \text{driving = fail} \)
- \( \text{driving = pass, theory = pass} \rightarrow \text{license = pass} \)
- \( \text{license = pass} \rightarrow \text{driving = pass, theory = pass} \)
- \( \text{license = fail, theory = fail, driving = pass, driving = fail} \rightarrow \bot \)
- \( \text{license = fail, theory = fail, theory = pass, driving = fail} \rightarrow \bot \)
“Difficult” Background Knowledge

We know beforehand: pass, fail $\rightarrow \bot$ and $\top \rightarrow$ pass or fail, i.e., the background knowledge contains a clause.

Def.: A clause is a pair of subsets

$$A, B \subseteq M, \text{ written as } A \rightarrow B$$

A clause holds in a formal context $(G, M, I)$, iff for all $g \in G$

$$A \subseteq g' \text{ implies } B \cap g' \neq \emptyset$$

i.e., every object that has all attributes from $A$ has at least one attribute from $B$
clauses are *more expressive and powerful* than implications

actually, *every* propositional formula is logically equivalent to a conjunction of clauses ("conjunctive normal form")

*deciding if a given clause follows from a given list of clauses is hard* ($\mathcal{NP}$-complete)

it is even $\mathcal{NP}$-complete to infer if a given *implication* can be inferred from a given list of clauses

thus: as easy it is to compute the stem base, to find a base for clauses is not so easy
Difficult Background Knowledge

Rep.: pseudo-intent, pseudo-closure

On the blackboard: Def. pseudo-model

On the blackboard: example
On the blackboard: **Def.** cumulated clause