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

Agenda

6. Background Knowledge
   - Simplifying Implications of the Stem Base
   - Optimizing the Computation of the Stem Base
   - Optimizing the Computation of the Stem Base
   - Incomplete Knowledge About Objects
   - “Harmless” Background Knowledge
   - “Difficult” Background Knowledge

Simplifying Implications of the Stem Base

- 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 $a \to b, c$ we can simplify $d, e \to a, b, c$ to $d, e \to a, b$
- *Several* redundant attributes can not always be removed:
  - a and c can not be removed from $d, e \to a, b, c$

exemplary stem base

$$a \to b, c$$
$$d, e \to a, b, c$$
$$c, e \to a, b, d$$
$$c, d \to a, b, e$$

Optimizing the Computation of the Stem Base

- Assume that we have computed the implication $\{c, d\} \to \{a, b, e\}$ (of the attribute set $\{a, b, c, d, e\}$):
  - then $\textsc{Next Closure}$ checks $\{c, d\} <_e \{a, b, c, d, e\}$ – which fails
  - *improvement*: directly continue with $i := b$
- Similar, after the implication $\{a\} \to \{b, c\}$:
  - $\textsc{Next Closure}$ is unsuccessfully checking $\{a\} <_e \{a, b, c, e\}$, $\{a\} <_d \{a, b, c, d\}$, $\{a\} <_c \{a, b, c\}$
  - *improvement*: directly continue with $A := \{a, b, c\}$

Optimizing the Computation of the Stem Base

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 $i < k$

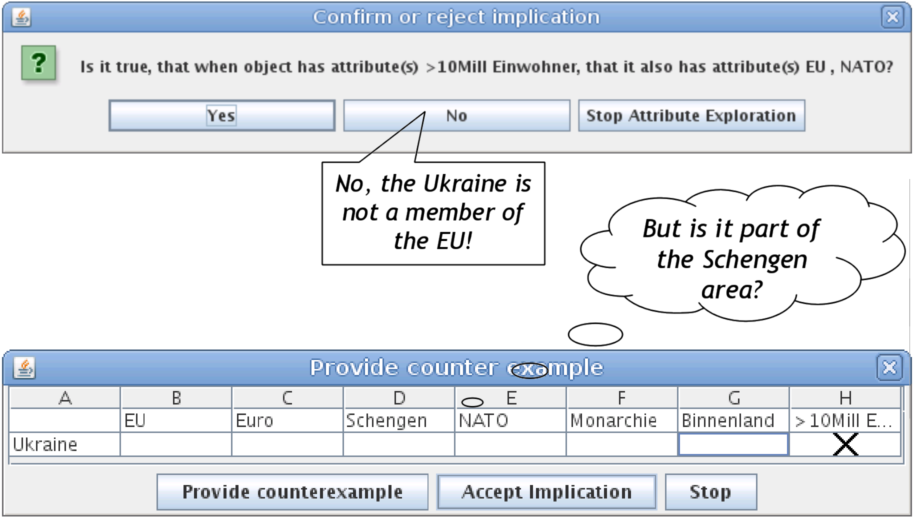
- In the example $\{c,d\} \to \{a,b,e\}$:
  $d = \max\{c,d\}, a = \min\{a,b,c,d,e\} \setminus \{c,d\}$ – ignore e and d

$k < l$: 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 lectically largest pseudo-intent with the closure P''.
- In the example $\{a\} \to \{b,c\}$ 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:



Incomplete Knowledge About Objects

- First, we start with a context $\mathbb{E} = (E, M, J)$ with *examples* E of a context (G, M, I) (i.e., $E \subseteq G$ and $J := I \cap (E \times M)$)
- Then, we replace $\mathbb{E}$ by $\mathbb{E}_+ = (E, M, J_+)$ and $\mathbb{E}_? = (E, M, J_?)$ with $J_+ \subseteq J \subseteq J_?$
- $(\mathbb{E}_+, \mathbb{E}_?)$ 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^+ und $e^? := M \setminus e^-$ are added to $\mathbb{E}_+$ and $\mathbb{E}_?$, respectively
- For $B := \mathcal{L}^\bullet(A + i)$ we compute instead of B'':

$$B^{+?} := \bigcap\{e^? \mid e \in E, B \subseteq e^+\}$$

  (that's not a closure operator!)
- Any modification of the list $\mathcal{L}$ leads to a modification of $\mathbb{E}_+$ and $\mathbb{E}_?$:
  - 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 $\mathcal{L}(e^+ \cup \{m\}) \subseteq e^?$

Incomplete Knowledge About Objects

- 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

	theory		driving		license	
	pass	fail	pass	fail	pass	fail
1	×		×		×	
2	×			×		×
3		×	×			×
4		×		×		×

The stem base for the context

$$\text{driving} = \text{fail} \rightarrow \text{license} = \text{fail}$$
$$\text{theory} = \text{fail} \rightarrow \text{license} = \text{fail}$$
$$\text{license} = \text{fail}, \text{driving} = \text{pass} \rightarrow \text{theory} = \text{fail}$$
$$\text{license} = \text{fail}, \text{theory} = \text{pass} \rightarrow \text{driving} = \text{fail}$$
$$\text{driving} = \text{pass}, \text{theory} = \text{pass} \rightarrow \text{license} = \text{pass}$$
$$\text{license} = \text{pass} \rightarrow \begin{array}{l}\text{driving} = \text{pass},\\ \text{theory} = \text{pass}\end{array}$$
$$\begin{array}{r}\text{license} = \text{fail}, \text{theory} = \text{fail},\\ \text{driving} = \text{pass}, \text{driving} = \text{fail}\end{array} \rightarrow \bot$$
$$\begin{array}{r}\text{license} = \text{fail}, \text{theory} = \text{fail},\\ \text{theory} = \text{pass}, \text{driving} = \text{fail}\end{array} \rightarrow \bot$$

Wouldn't we rather expect

$$\text{theory} = \text{pass}, \text{driving} = \text{pass} \leftrightarrow \text{license} = \text{pass}?$$

Background Knowledge

- This does not work, because we intuitively assume that “fail” is the negation of “pass”, i.e., we assume that

$$\text{pass, fail} \to \bot \text{ and } \top \to \text{pass or fail}$$

  hold as *background knowledge* for all parts of the test.

- *pass, fail* $\to \bot$ is an *implication*
- $\top \to$ *pass or fail* is a *clause*

“Harmless” Background Knowledge

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 *relative* to the manually added implications.
- If we add implications during exploration, the resulting set of implications is not necessarily redundant.

“Harmless” Background Knowledge

If we include *pass, fail* $\to \bot$ (i.e., the three implications “theory $=$ pass, theory $=$ fail $\to \bot$”, etc.) as background knowledge, we get a base with six implications:

	theory		driving		license	
	pass	fail	pass	fail	pass	fail
1	×		×		×	
2	×			×		×
3		×	×			×
4		×		×		×

$$\text{driving} = \text{fail} \to \text{license} = \text{fail}$$

$$\text{theory} = \text{fail} \to \text{license} = \text{fail}$$

$$\text{license} = \text{fail}, \text{driving} = \text{pass} \to \text{theory} = \text{fail}$$

$$\text{license} = \text{fail}, \text{theory} = \text{pass} \to \text{driving} = \text{fail}$$

$$\text{driving} = \text{pass}, \text{theory} = \text{pass} \to \text{license} = \text{pass}$$

$$\text{license} = \text{pass} \to \begin{array}{l}\text{driving} = \text{pass},\\ \text{theory} = \text{pass}\end{array}$$

$$\begin{array}{r}\text{license} = \text{fail}, \text{theory} = \text{fail},\\ \text{driving} = \text{pass}, \text{driving} = \text{fail}\end{array} \to \bot$$

$$\begin{array}{r}\text{license} = \text{fail}, \text{theory} = \text{fail},\\ \text{theory} = \text{pass}, \text{driving} = \text{fail}\end{array} \to \bot$$

"Difficult" Background Knowledge

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

Def.: A *clause* is a pair of subsets

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

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*

"Difficult" Background Knowledge

- 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* gan 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

“Difficult” Background Knowledge

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