Formal Concept Analysis Methods for Description Logics

Barış Sertkaya

Theoretical Computer Science
TU Dresden, Germany

November 14, 2007
1. Introduction
   - Formal Concept Analysis
   - Description Logics

2. Supporting bottom-up approach

3. Completing Description Logic knowledge bases

4. On the generators of closed sets
Formal Concept Analysis (FCA)

Branch of mathematics for conceptual data analysis and knowledge processing

- $\mathbb{K} = (G, M, I)$ a formal context, $G$ the objects, $M$ the attributes, $I$ the incidence relation

### Formal context

<table>
<thead>
<tr>
<th>$\mathbb{K}_{\text{countries}}$</th>
<th>has nuclear weapons</th>
<th>UN Security Council permanent</th>
<th>Council temporary</th>
<th>G8 mem.</th>
<th>EU mem.</th>
<th>UN mem.</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Germany</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>France</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>UK</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Turkey</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Qatar</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Italy</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
formal concepts are "natural clusterings" of the data
they form a complete lattice called the concept lattice
an implication $A \rightarrow B$ holds in $\mathbb{K}$ if every object having the
attributes in $A$ also has the attributes in $B$ ($A, B \subseteq M$)
Attribute exploration

Method for acquiring complete knowledge by asking questions to a domain expert

- a formal context \( K \), and a domain expert
- ask the expert: “is it true that objects with attributes \( a_1, \ldots, a_n \) also have the attributes \( b_1, \ldots, b_m \)?”
  - if yes: store the implication
  - if no: get a counterexample from the user, extend \( K \)

- the resultant knowledge is complete: every implication either follows from the stored implications, or is refuted by \( K \)
- asks the minimum number of questions
Description Logics

Logic-based knowledge representation formalisms

- Guitar $\equiv$ Instrument $\sqcap$ $\exists$ has-part. String
- AcousticGuitar $\equiv$ Guitar $\sqcap$ $\exists$ has-part. Resonator
- ElectricGuitar $\equiv$ Guitar $\sqcap$ $\neg$$\exists$ has-part. Resonator $\sqcap$ $\exists$ has-part. Pickup

- build concept descriptions from concept names and role names using constructors

  - $\mathcal{ALE}$: $\top$, $\bot$, $\neg A$, $\sqcap$, $\exists$, $\forall$
  - $\mathcal{ALC}$: $\top$, $\bot$, $\neg A$, $\neg C$, $\sqcap$, $\sqcup$, $\exists$, $\forall$

- TBox a set of concept definitions
- ABox a set of concept assertions, e.g. $C(a)$
Description Logics

Semantics w.r.t. an interpretation
\[ \mathcal{I} = (\Delta^I, \cdot^I) \]

- A model of a TBox \( \mathcal{T} \) satisfies all definitions in \( \mathcal{T} \)
- A model of an ABox \( \mathcal{A} \) satisfies all assertions in \( \mathcal{A} \)
- A KB \((\mathcal{T}, \mathcal{A})\) consistent if \( \mathcal{T} \) and \( \mathcal{A} \) have a common model
- Subsumption: \( C \sqsubseteq^\mathcal{T} D \) if \( \Delta^I_C \subseteq \Delta^I_D \) holds for all models of \( \mathcal{T} \)

| \( \top \) | \( \Delta^I \) |
| \( \bot \) | \( \emptyset \) |
| \( \neg C \) | \( \Delta^I_C \setminus \emptyset \) |
| \( C \sqcap D \) | \( C^I \cap D^I \) |
| \( C \sqcup D \) | \( C^I \cup D^I \) |
| \( \exists r.C \) | \( \{ x \in \Delta^I | \exists y : (x, y) \in r^I \land y \in C^I \} \) |
| \( \forall r.C \) | \( \{ x \in \Delta^I | \forall y : (x, y) \in r^I \rightarrow y \in C^I \} \) |
Supporting bottom-up construction

- **bottom-up approach**: generate a concept description from the provided examples
- Consider the scenario: a background KB $\mathcal{T}$ written in an expressive DL $\mathcal{L}_2$
- Allow user to extend it using a less expressive DL $\mathcal{L}_1$, and still support the bottom-up approach
- $\mathcal{L}_1$-concept descriptions that contain names defined in $\mathcal{T}$
- Compute common subsumer of such concept descriptions
Supporting bottom-up construction

\textit{ALC} TBox $\mathcal{T}$

\begin{align*}
\text{BassInstrument} &\equiv \text{Instrument} \sqcap \forall \text{produces-tone}.\text{Bass} \\
\text{AcousticBassGuitar} &\equiv \text{AcousticGuitar} \sqcap \forall \text{produces-tone}.\text{Bass} \\
\text{ElectricBassGuitar} &\equiv \text{ElectricGuitar} \sqcap \forall \text{produces-tone}.\text{Bass} \\
\text{BassGuitar} &\equiv \text{AcousticBassGuitar} \sqcup \text{ElectricBassGuitar}
\end{align*}

compute a common subsumer of

\begin{align*}
\exists \text{plays}.(\text{ElectricGuitar} \sqcap \text{BassInstrument}) \text{ and } \\
\exists \text{plays}.(\text{AcousticGuitar} \sqcap \text{BassInstrument})
\end{align*}

- if we ignore $\mathcal{T}$: $\exists \text{plays}.\text{BassInstrument}$
- if we take $\mathcal{T}$ into account: $\exists \text{plays}.\text{BassGuitar}$ (more specific result)
Supporting bottom-up construction

**$\mathcal{ALC}$ TBox $T$**

- $\text{BassInstrument} \equiv \text{Instrument} \sqcap \forall \text{produces-tone}.\text{Bass}$
- $\text{AcousticBassGuitar} \equiv \text{AcousticGuitar} \sqcap \forall \text{produces-tone}.\text{Bass}$
- $\text{ElectricBassGuitar} \equiv \text{ElectricGuitar} \sqcap \forall \text{produces-tone}.\text{Bass}$
- $\text{BassGuitar} \equiv \text{AcousticBassGuitar} \sqcup \text{ElectricBassGuitar}$

**compute a common subsumer of**

- $\exists \text{plays}.(\text{ElectricGuitar} \sqcap \text{BassInstrument})$ and $\exists \text{plays}.(\text{AcousticGuitar} \sqcap \text{BassInstrument})$

- if we ignore $T$: $\exists \text{plays}.\text{BassInstrument}$
- if we take $T$ into account: $\exists \text{plays}.\text{BassGuitar}$ (more specific result)
Using FCA

- subsumption hierarchy of conjunctions of (negated) concept names in the background KB $\mathcal{T}$
- in our scenario the background KB is fixed, precompute this hierarchy
- checking subsumption between each pair of conjunctions inefficient, $2^m \times 2^m$ subsumption tests ($m = |\mathcal{T}|$)
- use attribute exploration, guarantees the minimum number of subsumption tests
- define a formal context s.t. the concept lattice is isomorphic to this hierarchy
- a DL reasoner can act as expert
Using FCA

\( \mathcal{T} \) background ontology written in \( \mathcal{L}_2 \). Define \( \mathbb{K} = (G, M, I) \):

\[
G := \{ C \mid C \text{ an } \mathcal{L}_2\text{-concept description}\}, \\
M := \{ A_1, \ldots, A_n \} \text{ concept names in } \mathcal{T}, \\
I := \{(C, A) \mid C \subseteq_{\mathcal{T}} A\}.
\]

**Lemma:** \( B_1 \rightarrow B_2 \) iff \( \sqcap B_1 \sqsubseteq_{\mathcal{T}} \sqcap B_2 \)

(i.e., implications in \( \mathbb{K} \) correspond to subsumption relations in \( \mathcal{T} \))

**Theorem:** The concept lattice of \( \mathbb{K} \) is isomorphic to the desired hierarchy

- Supremum of two concept descriptions can efficiently be computed
- Implemented a prototype
Generalization of the method

- hierarchy of conjunctions of concept names in a TBox [Baa95]
- hierarchy of least common subsumers [BM00]
- can be generalized to compute the hierarchy of infima of a partially ordered set

\((P, \preceq)\) a partially ordered set, and \(N\) a finite subset of \(P\). Define \(K_\preceq = (G, M, I)\):

\[
G := P,
M := N,
I := \{(g, m) \mid g \preceq m \text{ for } g \in G, m \in M\}.
\]
Completing DL knowledge bases

When is an ontology complete?

1. Implications between classes of the application domain follow from the TBox
2. The ABox contains counterexamples to all other implications

Does my ontology contain everything I want?

detect “missing”
- relations between classes
- individuals

extend the ontology appropriately
Completing DL knowledge bases

- detect “missing” relations between classes
- individuals
- extend the ontology appropriately

When is an ontology complete?

1. Implications between classes of the application domain follow from the TBox
2. The ABox contains counterexamples to all other implications
Completeness of a DL KB

\((\mathcal{T}, \mathcal{A})\) a consistent DL knowledge base, \(M\) a set of concept descriptions and \(L, R \subseteq M\), and \(\mathcal{I}\) a model of \((\mathcal{T}, \mathcal{A})\).

\((\mathcal{T}, \mathcal{A})\) is \textit{complete} w.r.t. \(\mathcal{I}\) if the following are equivalent:

1. \(L \rightarrow R\) holds in \(\mathcal{I}\)
2. \(L \rightarrow R\) follows from \(\mathcal{T}\)
3. \((\mathcal{T}, \mathcal{A})\) does not contain a counterexample to \(\sqcap L \sqsubseteq \sqcap R\)

- \(2^k \times 2^k\) implications to check (\(k = |M|\))
- many of them trivial
- do not bother the domain expert redundantly
- a smart way to organize these questions
Completeness of a DL KB

(\mathcal{T}, \mathcal{A}) \text{ a consistent DL knowledge base, } M \text{ a set of concept descriptions and } L, R \subseteq M, \text{ and } \mathcal{I} \text{ a model of } (\mathcal{T}, \mathcal{A}).

(\mathcal{T}, \mathcal{A}) \text{ is complete w.r.t. } \mathcal{I} \text{ if the following are equivalent:}

1. \( L \rightarrow R \) holds in \( \mathcal{I} \)
2. \( L \rightarrow R \) follows from \( \mathcal{T} \)
3. \( (\mathcal{T}, \mathcal{A}) \) does not contain a counterexample to \( \sqcap L \sqsubseteq \sqcap R \)

- \( 2^k \times 2^k \) implications to check (\( k = |M| \))
- many of them trivial
- do not bother the domain expert redundantly
- a smart way to organize these questions
Completion process

when a question is asked

- first check if it follows from the TBox
- if not ask the expert
- if the expert confirms, add a new axiom to the TBox
- if the expert has a counterexample, add it to the ABox
FCA with partial knowledge

- Classical FCA assumes closed world
- To use with DL KBs, we need an extension to open world
- A partial object \((A, S)\) s.t. \(A, S \subseteq M\) and \(A \cup S = \emptyset\)
- A partial context a set of partial objects
- \((A, S)\) refutes an implication \(L \rightarrow R\) if \(L \subseteq A\) and \(R \cup S \neq \emptyset\)
- Attribute exploration that works with partial knowledge
- Proved correctness and minimum number of questions to the expert
- The expert can give partial objects as counterexample
An ABox individual $a$ induces a partial object $(A, S)$ s.t.

\[
A = \{ C \in M | T, A \models C(a) \} \text{ and } S = \{ C \in M | T, A \models \neg C(a) \}
\]

An ABox induces a partial context $(\mathcal{T}, \mathcal{A})$ refutes $L \rightarrow R$ if $(\mathcal{T}, \mathcal{A})$ contains a counterexample to $\sqcap L \subseteq \sqcap R$

$L \rightarrow R$ follows from $\mathcal{T}$ if $\sqcap L \sqsubseteq_{\mathcal{T}} \sqcap R$

Attribute exploration for use with DL KBs

Proved correctness and minimum number of questions

Prototype implementation as Swoop extension, talks OWL
Generators of implication-closed sets

\( \mathcal{L} \) a set of implications on \( M, \ P \subseteq M \) closed under \( \mathcal{L} \), and \( Q \subseteq M \)

- \( Q \) a **generator** of \( P \) if \( \mathcal{L}(Q) = P \)

- \( Q \) a **minimal generator** of \( P \) if \( \mathcal{L}(Q) = P \), and \( Q \) minimal w.r.t. set inclusion

- \( Q \) a **minimum cardinality generator** of \( P \) if \( \mathcal{L}(Q) \setminus Q = P \), and \( Q \) minimal w.r.t. cardinality

<table>
<thead>
<tr>
<th>Is there a generator of cardinality less than ( m )?</th>
<th>NP-complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find the number of minimal generators</td>
<td>#P-complete</td>
</tr>
<tr>
<td>Find the number of minimum cardinality generators</td>
<td>#·coNP-complete</td>
</tr>
</tbody>
</table>
Generators of implication-closed sets

\( \mathcal{L} \) a set of implications on \( M \), \( P \subseteq M \) closed under \( \mathcal{L} \), and \( Q \subseteq M \)

- \( Q \) a generator of \( P \) if \( \mathcal{L}(Q) = P \)
- \( Q \) a minimal generator of \( P \) if \( \mathcal{L}(Q) = P \), and \( Q \) minimal w.r.t. set inclusion
- \( Q \) a minimum cardinality generator of \( P \) if \( \mathcal{L}(Q) \setminus Q = P \), and \( Q \) minimal w.r.t. cardinality

<table>
<thead>
<tr>
<th>Question</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is there a generator of cardinality less than ( m )?</td>
<td>NP-complete</td>
</tr>
<tr>
<td>Find the number of minimal generators</td>
<td>#P-complete</td>
</tr>
<tr>
<td>Find the number of minimum cardinality generators</td>
<td>#·coNP-complete</td>
</tr>
</tbody>
</table>
Connection to relational databases

\( \langle R, F \rangle \) a relation scheme with attribute set \( R \), and a set of functional dependencies \( F \)

- \( K \subseteq R \) a \textbf{key} of \( \langle R, F \rangle \) if \( K \rightarrow R \in F^+ \)
- \( K \) a \textbf{minimal key} if it is a key, and minimal w.r.t set inclusion

<table>
<thead>
<tr>
<th>Question</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is there a key of cardinality less than ( m )?</td>
<td>NP-complete</td>
</tr>
<tr>
<td>Find the number of minimal keys</td>
<td>#P-complete</td>
</tr>
</tbody>
</table>

[LO78] 
[GKM+03]
< R, F > a relation scheme with attribute set R, and a set of functional dependencies F

- K ⊆ R a key of < R, F > if K → R ∈ F^+
- K a minimal key if it is a key, and minimal w.r.t set inclusion

<table>
<thead>
<tr>
<th>Is there a key of cardinality less than m?</th>
<th>NP-complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find the number of minimal keys</td>
<td>#P-complete</td>
</tr>
</tbody>
</table>

[LO78] [GKM+03]
Finding all minimal generators

**Problem**: Given the implications $\mathcal{L}$ on $M$ and $P \subseteq M$ closed under $\mathcal{L}$, find all minimal generators of $P$

- Finding one minimal generator can be done in $O(|\mathcal{L}||M|^2)$
- Finding all minimal generators is equivalent to finding all minimal keys of a relation scheme
- All minimal keys can be found in incremental polynomial time
  - [LO78]
- Modification of this algorithm for finding all minimal generators
\( K = (G, M, I) \) a formal context, \( P \subseteq M \) a concept intent, and \( Q \subseteq P \)

- \( Q \) a generator of \( P \) if \( Q'' = P \)
- \( Q \) a minimal generator of \( P \) if \( Q'' = P \), and \( Q \) minimal w.r.t. set inclusion

<table>
<thead>
<tr>
<th>Is there a generator with cardinality less than or equal to ( m )?</th>
<th>NP-complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find the number of minimal generators</td>
<td>( #P )-complete</td>
</tr>
</tbody>
</table>
Generators of concept intents

\[ K = (G, M, I) \] a formal context, \( P \subseteq M \) a concept intent, and \( Q \subseteq P \)

- \( Q \) a generator of \( P \) if \( Q'' = P \)
- \( Q \) a minimal generator of \( P \) if \( Q'' = P \), and \( Q \) minimal w.r.t. set inclusion

<table>
<thead>
<tr>
<th>Is there a generator with cardinality less than or equal to ( m )?</th>
<th>NP-complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find the number of minimal generators</td>
<td>#P-complete</td>
</tr>
</tbody>
</table>
F. Baader.
Computing a Minimal Representation of the Subsumption Lattice of all Conjunctions of Concepts Defined in a Terminology. 1995

F. Baader and R. Molitor.
Building and Structuring Description Logic Knowledge Bases Using Least Common Subsumers and Concept Analysis. 2000

Discovering all most specific sentences. 2003

C. L. Lucchesi and S. L. Osborn.
Candidate Keys for Relations. 1978