Enhancing OWL Ontologies via Formal Concept Analysis

http://sites.google.com/site/ijcai11fcaontologytutorial

INTRODUCTION

Sebastian Rudolph and Barış Sertkaya
This Tutorial – Presenters

- **Sebastian Rudolph**
  - Institut AIFB, Karlsruhe Institute of Technology
  - PhD in Mathematics (TU Dresden)
  - [http://www.sebastian-rudolph.de](http://www.sebastian-rudolph.de)

- **Barış Sertkaya**
  - SAP Research Center Dresden
  - PhD in Computer Science (TU Dresden)
  - [http://sites.google.com/site/sertkayabar]s

Tutorial Homepage:
[http://sites.google.com/site/ijcai11fcaontologytutorial](http://sites.google.com/site/ijcai11fcaontologytutorial)
This Tutorial – Schedule

14:30-14:45 Welcome and introduction
14:45-15:15 Theoretical foundations of FCA
15:15-15:45 Introduction to attribute exploration
15:45-16:30 Introduction to OWL
16:30-17:00 Coffee break
17:00-17:45 Applications of attribute exploration to ontologies
17:45-18:30 Software demonstration and hands-on session
Formal Concept Analysis?

- Branch of Applied Mathematics
- Based on **Lattice Theory** developed by Garrett Birkhoff and others in the 1930s
- Employs algebra in order to formalize notions of **concept** and **conceptual hierarchy**
- Term **Formal Concept Analysis** (short: FCA) introduced by Rudolf Wille in the 1980s.
Why Formal Concept Analysis?

- The method of Formal Concept Analysis offers an algebraic approach to data analysis and knowledge processing.

- Strengths of FCA are
  - ... a solid mathematical and philosophical foundation,
  - ... more than 1000 research publications,
  - ... experience of several hundred application projects,
  - ... an expressive and intuitive graphical representation,
  - and a good algorithmic basis.

- Due to its elementary yet powerful formal theory, FCA can express other methods, and therefore has the potential to unify the methodology of data analysis.
FCA – Further Information

- Conferences
  - International Conference on Formal Concept Analysis (ICFCA)
    next: May 2012, Leuven, Belgium
  - International Conference on Conceptual Structures (ICCS)
    next: July 2011, Derby, UK
    http://www.iccs.info
  - Concept Lattices and Applications (CLA)
    next: October 2011, Nancy, France

- Monograph
  - Bernhard Ganter & Rudolf Wille.
    „Formal Concept Analysis. Mathematical Foundations“
    Springer Verlag, 1999

- FCA Website by Uta Priss: http://www.upriss.org.uk/fca/fca.html
Acknowledgements

- Slides of this tutorial are partially based on material from ...
  - Johanna Völker,
  - Peter Becker,
  - Bernhard Ganter,
  - Gerd Stumme, and
  - Bastian Wormuth.
THANK YOU.

Sebastian Rudolph and Barış Sertkaya
 Enhancing OWL Ontologies via Formal Concept Analysis

https://sites.google.com/site/ijcai11fcaontologytutorial/

Sebastian Rudolph and Barış Sertkaya

16.07.2011

FOUNDATIONS OF FORMAL CONCEPT ANALYSIS
Introduction

- Formal Concept Analysis (FCA) is a...
  - "mathematization" of the philosophical understanding of concepts
  - human-centered method to structure and analyze data
  - method to visualize data and its inherent structures, implications and dependencies

The following slides are partially based on the ICFCA'04 tutorial by Bastian Wormuth and Peter Becker
http://www.wormuth.info/ICFCA04/materials.html
What is a Concept?

Consider the concept “bird”. What drives us to call something a “bird”?

Every object with certain attributes is called “bird”:

- A bird has feathers.
- A bird has two legs.
- A bird has a bill. ...

All objects having these attributes are called “birds”:

- Duck, goose, owl and parrot are birds.
- Penguins are birds, too.
- ...

What is a Concept?

- This description of the concept “bird” is based on sets of

  objects related to attributes.

  duck
goose
parrot
...

  has bill
  has feathers
  has two legs
...

→ Objects, attributes and a relation form a **formal concept**.
What is a Concept?

- So, a formal concept is constituted by two parts

  - A: a set of objects
  - B: a set of attributes

- ... having a certain relation:
  - every object belonging to this concept has all the attributes in B
  - every attribute belonging to this concept is shared by all objects in A

- A is called the concept's extent, B is called the concept's intent
A repertoire of **objects** and **attributes** (which might or might not be related) constitutes the „context“ of our considerations.
# The Formal Context

A formal context is a set of objects (G) and a set of attributes (M) with an incidence relation (I) between them. The incidence relation indicates which objects have which attributes.

<table>
<thead>
<tr>
<th>K</th>
<th>small</th>
<th>medium</th>
<th>big</th>
<th>2legs</th>
<th>4legs</th>
<th>feathers</th>
<th>hair</th>
<th>fly</th>
<th>hunt</th>
<th>run</th>
<th>swim</th>
<th>mane</th>
<th>hooves</th>
</tr>
</thead>
<tbody>
<tr>
<td>dove</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>hen</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>
<td></td>
<td></td>
</tr>
<tr>
<td>duck</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>goose</td>
<td>x</td>
<td></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>owl</td>
<td>x</td>
<td></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>hawk</td>
<td>x</td>
<td></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>eagle</td>
<td></td>
<td></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>fox</td>
<td></td>
<td></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>dog</td>
<td></td>
<td></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>wolf</td>
<td></td>
<td></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>cat</td>
<td></td>
<td></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>tiger</td>
<td></td>
<td></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>lion</td>
<td></td>
<td></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>horse</td>
<td></td>
<td></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>zebra</td>
<td></td>
<td></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>cow</td>
<td></td>
<td></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>
</tbody>
</table>

**set of attributes (M)**

**set of objects (G)**

 crosses indicate incidence relation (I) between G and M

(G,M,I) is called formal context
Definition of Formal Concepts

- For the mathematical definition of *formal concepts* we introduce the *derivation operator* “‘”.

For a set of objects $A$, $A'$ is defined as:
$$A' = \{\text{all attributes in } M \text{ common to the objects of } A\}$$

For a set of attributes $B$, $B'$ is defined as:
$$B' = \{\text{objects in } G \text{ having all attributes of } B\}$$

- We are looking for pairs $(A, B)$ of objects $A$ and attributes $B$ that satisfy the conditions
$$A' = B \text{ and } B' = A$$
and we call these pairs *formal concepts*. 
Calculating Formal Concepts

Using the derivation operator we can derive formal concepts from our formal context with the following procedure:

- Pick an object set $A$.
- Derive the attributes $A'$.
- Derive $(A')'$.
- $(A'',A')$ is a formal concept.

The same routine could be applied starting with an attribute set $B$: $(B',B'')$ is a formal concept as well.
Calculating Formal Concepts

1. Pick a set of objects: \( A = \{ \text{duck} \} \)
2. Derive attributes: \( A' = \{ \text{small, 2legs, feathers, fly, swim} \} \)
3. Derive objects: \( (A')' = \{ \text{small, 2legs, feathers, fly, swim} \} = \{ \text{duck, goose} \} \)
4. Formal concept: \( (A'', A') = (\{ \text{duck, goose} \}, \{ \text{small, 2legs, feathers, fly, swim} \}) \)
The **formal concept** \((A'',A')=\{(\text{duck, goose}), \{\text{small, 2legs, feathers, fly, swim}\}\) is represented in the line diagram as a node:

\[
\begin{align*}
B' &= \{\text{duck, goose, dove, owl, hawk}\} \\
B'' &= \{\text{small, 2legs, feathers, fly}\}
\end{align*}
\]

Consider another **formal concept** \((B',B'')=\{(\text{duck, goose, dove, owl, hawk}), \{\text{small, 2legs, feathers, fly}\}\).

The **formal concept** \((B',B'')\) is a **superconcept** of \((A'',A')\) and \((A'',A')\) is a **subconcept** of \((B',B'')\), because \(A''\) is a subset of \(B'\).

So \((B',B'')\) is drawn **above** \((A'',A')\) and connected to it by a line.
Ordering Concepts

We extend the diagram by adding more **formal concepts**
({owl, hawk}, {feathers, 2legs, small, fly, hunt})
({owl, hawk, eagle}, {feathers, 2legs, fly, hunt})

... and **subconcept relations:**

... and so on.

Several methods exist to derive all **formal concepts:**
Cut over extents, Ganter's algorithm etc.
The Concept Lattice

- The subconcept–superconcept relation defines an order $\subseteq$ on the set $B$ of all formal concepts of a formal context.
- For two concepts $(A_1, A_2)$ and $(B_1, B_2)$, this order is defined by:
  $$(A_1, A_2) \subseteq (B_1, B_2) \iff A_1 \subseteq B_1 (\iff B_2 \subseteq A_2)$$
- $(A_1, A_2)$ is smaller than $(B_1, B_2)$ if $A_1$ is subset of $B_1$ (objects) and $B_2$ is subset of $A_2$ (attributes). Hence, $(B, \subseteq)$ is an ordered set.
- The set $B$ of formal concepts has another property:
  - For each family of formal concepts of a formal context there exists always a unique greatest subconcept and a unique smallest superconcept.
  - The ordered set $B = (B, \subseteq)$ plus the last property forms a mathematical structure: the concept lattice.
Concept Lattice – Formal Concepts

\{duck, goose, dove, owl, hawk, eagle\}
\{2legs, feathers, fly\}

\{duck, goose, dove, owl, hawk\}
\{small, 2legs, feathers, fly\}

\{eagle\}
\{medium, hunt, 2legs, feathers, fly\}

\{duck, goose\}
\{small, swim, 2legs, feathers, fly\}
Concept Lattice – Top and Bottom

more general concepts

more specific concepts

all attributes

no objects

no attributes

all objects
How to determine all the implications for a given lattice or context?
Summary

- Formal contexts
  - Objects, attributes and incidence relation
- ... and formal concepts
  - Extent and intent
  - Subconcept relations
- Concept lattices
  - How to interpret a concept lattice
  - Generalization and specialization
  - Implications
- Next: attribute exploration
THANK YOU.

Sebastian Rudolph and Barış Sertkaya
Enhancing OWL Ontologies via Formal Concept Analysis
https://sites.google.com/site/ijcai11fcaontologytutorial/

ATTRIBUTE EXPLORATION

16.07.2011  Sebastian Rudolph and Barış Sertkaya
Attribute Implications (aka propositional Horn clauses)

- For $A, B \subseteq M$, the implication $A \rightarrow B$ holds in $K$, if every object having all attributes from $A$ also has all attributes from $B$.
- Formally: $A \subseteq \{g\}'$ implies $B \subseteq \{g\}'$ for all $g \in G$

- Examples:
  - $\{\text{wet}\} \rightarrow \{\text{fluid}\}$
  - $\{\text{fluid, dry}\} \rightarrow \{\text{warm}\}$
  - $\{\text{dry, wet}\} \rightarrow \{\text{cold}\} \ (\!)$
How to „Datamine“ Implications?

- We want to extract the „implicational“ knowledge from a formal context.
- Very naive approach: enumerate all \(2^{2|\mathcal{M}|}\) implications and check against context.
  - Takes way too long.
  - Generated implication set is extremely redundant.
- Examples:
  - \{fluid, dry\} → \{fluid\}
  - \{wet\} → \{fluid\} vs.
    - \{wet, cold\} → \{fluid\}

<table>
<thead>
<tr>
<th></th>
<th>fluid</th>
<th>dry</th>
<th>wet</th>
<th>warm</th>
<th>cold</th>
</tr>
</thead>
<tbody>
<tr>
<td>Earth</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>Water</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>Air</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fire</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td>×</td>
</tr>
</tbody>
</table>
How to „Datamine“ Implications?

- Observations:
  - For any attribute set $A$, the implication $A \rightarrow A''$ holds in $\mathbb{K}$
  - If $A \rightarrow B$ holds in $\mathbb{K}$ then $B \subseteq A''$

- Hence the implications of the form $A \rightarrow A''$ provide enough information about all implications of the context.

- Still rather naive approach: enumerate all $(2^{|M|})$ attribute sets $A$ and generate implication $A \rightarrow A''$
  - Still takes way too long
  - Generated implication set is still extremely redundant
Implication Bases

- Given a formal context $K$, a set of implications $\mathcal{S}$ is called *implication base of* $K$, if ...
  - every implication $A \rightarrow B$ from $\mathcal{S}$ holds in $K$,
  - every implication $A \rightarrow B$ holding in $K$ can be derived from $\mathcal{S}$, and
  - none of the implications from $\mathcal{S}$ can be derived from the other implications contained in $\mathcal{S}$

- Question: which $A \rightarrow A''$ to choose to make up an implication base?
The Stem Base

- Question: which \( A \rightarrow A'' \) to choose to make up an implication base?
- Answer: take all the pseudo-intents of \( K \).
- Attribute set \( P \) is called pseudo-intent, if
  - \( P \) is not an intent (i.e. \( P \neq P'' \)), but
  - if \( P \) contains another pseudo-intent \( Q \), then it also contains \( Q'' \)
- Definition recursive (but OK at least for finite \( M \))
- Set \( \{P \rightarrow P'' \mid P \text{ pseudo-intent}\} \) is called stem base
How to Compute the Stem Base

- We order attributes in a row:
  - e.g. $a,b,c,d,e,f$
- Based on that order, attribute sets are encoded as bit-vectors of length $|M|$:
  - e.g. $\{a,c,d\}$ becomes $[1,0,1,1,0,0]$
- Implications are pairs of bit-vectors:
  - e.g. $\{a\} \rightarrow \{a,e,f\}$ becomes $([1,0,0,0,0,0], [1,0,0,0,1,1])$
- Implications can be ,,applied“ to attribute sets:
  - $\{\{a\} \rightarrow \{a,e,f\}\}$ applied to $\{a,c,d\}$ yields $\{a,c,d,e,f\}$
    $([1,0,0,0,0,0], [1,0,0,0,1,1]) [1,0,1,1,0,0] = [1,0,1,1,1,1]$
- Implication sets can be applied to attribute sets:
  - $\{\{b,d\} \rightarrow \{c\}, \{a\} \rightarrow \{d\}\}$ applied to $\{a,b\}$ yields $\{a,b,c,d\}$
  - write $\mathcal{S}(A)$ for the result of applying implication set $\mathcal{S}$ to attribute set $A$
- $A+i$ defined as: take $A$, set $i$th bit to 1 and all subsequent bits to 0
  - e.g. $[0,1,0,0,1,1] + 3 = [0,1,1,0,0,0]$
How to Compute the Stem Base

- Input formal context $\mathcal{K}$
- Create list $\mathcal{S}$ of implications, initially empty
  Let $A = [0,0,...,0]$ (bit representation of empty set)
- Repeat
  - Add $A \rightarrow A''$ to $\mathcal{S}$ in case $A \neq A''$
  - Starting from $i = |M| + 1$, decrement $i$ until
    - $i=0$ or
    - The $i$th bit of $A$ is 0 and applying $\mathcal{S}$ to $A+i$ produces 1s only at positions greater than $i$
  - If $i=0$ output $\mathcal{S}$ and exit
  - Let $A = \mathcal{S}(A+i)$

... i ...

A: [0,0,1,0,1,1,0]
A+i: [0,0,1,1,0,0,0]
$\mathcal{S}(A+i)$: [0,0,1,1,0,1,1]
Interactive Knowledge Acquisition via Attribute Exploration

- Sometimes, $K$ is not entirely known from the beginning, but implicitly present as an expert’s knowledge.

- Attribute exploration determines the stembase of $K$ by asking expert for missing information.
Interactive Knowledge Acquisition via Attribute Exploration

- Sometimes, $K$ is not entirely known from the beginning, but implicitly present as an expert’s knowledge.
- Attribute exploration determines the stembase of $K$ by asking expert for missing information:
  - $M$ known and fixed
  - $H \subseteq G$ objects that are known in advance (as well as their attributes)
- Idea: use stembase algorithm on incomplete context which is updated on the fly
Stem Base Algorithm Revisited

- Input formal context $K=(H,M,J)$ where $J=(H \times M) \cap I$
- Create list $\mathcal{S}$ of implications, initially empty
  Let $A = [0,0,\ldots,0]$ (bit representation of empty set)
- Repeat
  - Add $A \rightarrow A''$ to $\mathcal{S}$ in case $A \neq A''$
  - Starting from $i = |M|+1$, decrement $i$ until
    - $i=0$ or
    - The $i$th bit of $A$ is 0 and applying $\mathcal{S}$ to $A+i$ produces 1s only at positions greater than $i$
  - If $i=0$ output $\mathcal{S}$ and exit
  - Let $A = \mathcal{S}(A+i)$

Has to be altered, because implication valid in $K$ might be invalid in $K$ since refuted by an object not yet recorded. Then augmenting $K$ by this object allows to refine the hypothesis.
Making It Interactive...

- Instead of just adding $A \rightarrow A''$ to $\emptyset$, do the following control loop:
  - While $A \neq A''$
    - Ask expert whether $A \rightarrow A''$ is valid in $K$
    - If yes, add $A \rightarrow A''$ to $\emptyset$ and exit while-loop, otherwise ask for counterexample and add it to $K$

- What is a counterexample for $A \rightarrow A''$?
  - An object having all attributes from $A$ but missing some from $A''$

- How to add a counterexample $g$ to $K=(H,M,J)$?
  - $H_{\text{new}} = H \cup \{g\}$
  - $J_{\text{new}} = J \cup \{(g,m) \mid m \text{ is attribute of } g \text{ in } K\}$
  - Essentially: just add a line to the cross table
Making It Interactive...

- Instead of just adding $A \rightarrow A''$ to $\mathcal{S}$, do the following control loop:
  - While $A \neq A'$,
    - Ask expert whether $A \rightarrow A''$ is valid in $K$
    - If yes, add $A \rightarrow A''$ to $\mathcal{S}$ and exit while-loop, otherwise ask for counterexample $g$ and add it to $K$

- Remarks:
  - Attribute set of $g$ has to comply with the implications already confirmed
  - Changing $K$ changes the operator $(.)''$
  - It is not obvious (but has to be proven) that this indeed works, i.e. the enumeration done beforehand is not corrupted by updating the context
Stem Base Algorithm Revisited

- Input formal context $\mathcal{K}$
- Create list $\mathcal{S}$ of implications, initially empty
  Let $A = [0,0,...,0]$  (bit representation of empty set)
- Repeat
  - While $A \neq A'$,
    - Ask expert whether $A \rightarrow A''$ is valid in $\mathcal{K}$
    - If yes, add $A \rightarrow A''$ to $\mathcal{S}$ and exit while-loop, otherwise ask for counterexample $g$ and add it to $\mathcal{K}$
  - Starting from $i = |M|+1$, decrement $i$ until
    - $i=0$ or
    - The $i$th bit of $A$ is 0 and
      applying $\mathcal{S}$ to $A+i$ produces 1s only at positions greater than $i$
  - If $i=0$ output $\mathcal{S}$ and exit
  - Let $A = \mathcal{S}(A+i)$
Extensions of Classical Attribute Exploration

- Allow for a-priori implications
  - Notion of relative stem base (Stumme 1996)

- Allow for arbitrary propositional background knowledge
  - Notion of frame context (Ganter 1999)

- Allow for partial description of objects
  - Notion of partial context (Burmeister, Holzer 2005)

- Allow for complete exploration of non-propositional logics
  - Horn logic with bounded variables: rule exploration (Zickwolflf 1991)
  - DLs with bounded role depth: relational exploration (Rudolph 2004)
THANK YOU.

Sebastian Rudolph and Barış Sertkaya
References I


References II


Enhancing OWL Ontologies via Formal Concept Analysis

https://sites.google.com/site/ijcai11fcaontologytutorial/

FOUNDATIONS OF OWL

16.07.2011

Sebastian Rudolph and Barış Sertkaya
What OWL Talks About (Semantics)

- both OWL 1 DL and OWL 2 DL are based on description logics
- here, we will treat OWL from the “description logic viewpoint“:
  - we use DL syntax
  - we won’t talk about datatypes and non-semantic features of OWL
- OWL (DL) ontologies talk about worlds that contain

- individuals
  - constants: pascal, anne

- classes / concepts
  - unary predicates: male(_), female(_)

- properties / roles
  - binary predicates: married( _, _ )
Assertional Knowledge

- asserts information about concrete named individuals
  - class membership: Male(Pascal)
    - rule version: \( \rightarrow \text{Male(pascal)} \)
  - property membership: married(anne,pascal)
    - rule version: \( \rightarrow \text{married (anne,pascal)} \)

That’s all what can be said with RDF!
Terminological Knowledge – Subclasses and Subproperties

- information about how classes and properties relate in general

- **subclass**: Child $\subseteq$ Person

  
  
  ```
  <owl:Class rdf:about="Woman">
    <rdfs:subClassOf rdf:resource="Person"/>
  </owl:Class>
  
  rule version: Child(x) $\rightarrow$ Person(x)
  ```

- **subproperty**: sonOf $\subseteq$ childOf

  
  ```
  <owl:ObjectProperty rdf:about="sonOf">
    <rdfs:subPropertyOf rdf:resource="childOf"/>
  </owl:ObjectProperty>
  
  rule version: sonOf(x,y) $\rightarrow$ childOf(x,y)
  ```
Class Constructors

- build new classes from class, property and individual names

  - union: `Actor ⊔ Politician`

    ```xml
    <owl:unionOf rdf:parseType="Collection">
      <owl:Class rdf:about="Actor"/>
      <owl:Class rdf:about="Politician"/>
    </owl:intersectionOf>
    ```

  - intersection: `Actor ⊓ Politician`

    ```xml
    <owl:intersectionOf rdf:parseType="Collection">
      <owl:Class rdf:about="Actor"/>
      <owl:Class rdf:about="Politician"/>
    </owl:intersectionOf>
    ```
Class Constructors

- build new classes from class, property and individual names
  
  - complement: \( \neg \text{Politician} \)
    
    \[
    \text{<owl:complementOf}
    \text{rdf:resource="Politician"}>
    \]
  
  - closed classes: \{anne, merula, pascal\}
    
    \[
    \text{<owl:oneOf rdf:parseType="Collection"}>
    \text{<rdf:Description rdf:about="anne"/>}
    \text{<rdf:Description rdf:about="merula"/>}
    \text{<rdf:Description rdf:about="pascal"/>}
    \text{</owl:oneOf>}
    \]
Class Constructors

- build new classes from class, property and individual names
  - existential quantification: \( \exists \text{hasChild.Male} \)

```xml
<owl:Restriction>
  <owl:onProperty rdf:resource="hasChild"/>
  <owl:someValuesFrom rdf:resource="Male"/>
</owl:Restriction>
```
Class Constructors

- build new classes from class, property and individual names
  - universal quantification: $\forall$hasChild.Female

```xml
<owl:Restriction>
  <owl:onProperty rdf:resource="hasChild"/>
  <owl:allValuesFrom rdf:resource="Female"/>
</owl:Restriction>
```
Class Constructors

- build new classes from class, property and individual names
  - cardinality restriction: ≥2hasChild.Female

```xml
<owl:Restriction>
  <owl:minQualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">2</owl:minQualifiedCardinality>
  <owl:onProperty rdf:about="hasChild"/>
  <owl:onClass rdf:about="Female"/>
</owl:Restriction>
```
Class Constructors

- build new classes from class, property and individual names
  - Self-restriction: \( \exists \text{killed}.\text{Self} \)

```xml
<owl:Restriction>
  <owl:onProperty rdf:resource="killed"/>
  <owl:hasSelf rdf:datatype="&xsd:boolean">true</owl:hasSelf>
</owl:Restriction>
```
Special Classes and Properties

- **special classes**
  - **top class**: $\top$
    ...class containing all individuals of the domain
    `<owl:Thing>`
  - **bottom class**: $\bot$
    ..."empty" class containing no individuals
    `<owl:Nothing>`

- **universal property**: $U$
  ...property linking every individual to every individual
  `<owl:topObjectProperty>`
Property Chain Axioms

- allow to infer the existence of a property from a chain of properties:

    - hasParent ∘ hasParent ⊆ hasGrandparent

      rule version: hasParent(x,y) ∧ hasParent(y,z) → hasGrandparent(x,z)

```
<rdf:Description rdf:about="hasGrandparent">
    <owl:propertyChainAxiom rdf:parseType="Collection">
        <owl:ObjectProperty rdf:about="hasParent"/>
        <owl:ObjectProperty rdf:about="hasParent"/>
    </owl:propertyChainAxiom>
</rdf:Description>
```
Property Chain Axioms

- allow to infer the existence of a property from a chain of properties:
  - hasEnemy $\circ$ hasFriend $\subseteq$ hasEnemy
  - rule version: hasEnemy(x,y) $\land$ hasFriend(y,z) $\rightarrow$ hasEnemy(x,z)

```
<rdf:Description rdf:about="hasEnemy">
  <owl:propertyChainAxiom rdf:parseType="Collection">
    <owl:ObjectProperty rdf:about="hasEnemy"/>
    <owl:ObjectProperty rdf:about="hasFriend"/>
  </owl:propertyChainAxiom>
</rdf:Description>
```
Property Chain Axioms: Caution! (1/2)

- arbitrary property chain axioms lead to undecidability
- restriction: set of property chain axioms has to be regular
  - there must be a strict linear order $\prec$ on the properties
  - every property chain axiom has to have one of the following forms:
    \[
    R \circ R \subseteq R \quad R^\prec \subseteq R \quad S_1 \circ S_2 \circ \ldots \circ S_n \subseteq R \\
    R \circ S_1 \circ S_2 \circ \ldots \circ S_n \subseteq R \\
    S_1 \circ S_2 \circ \ldots \circ S_n \circ R \subseteq R \\
    \]
  - thereby, $S_i \prec R$ for all $i=1, 2, \ldots, n$.

- Example 1: \[ R \circ S \subseteq R \quad S \circ S \subseteq S \quad R \circ S \circ R \subseteq T \]
  \[ \rightarrow \text{regular with order } S \prec R \prec T \]
- Example 2: \[ R \circ T \circ S \subseteq T \]
  \[ \rightarrow \text{not regular because form not admissible} \]
- Example 3: \[ R \circ S \subseteq S \quad S \circ R \subseteq R \]
  \[ \rightarrow \text{not regular because no adequate order exists} \]
combining property chain axioms and cardinality constraints may lead to undecidability

restriction: use only *simple* properties in cardinality expressions (i.e. those which cannot be – directly or indirectly – inferred from property chains)

technically:
- for any property chain axiom $S_1 \circ S_2 \circ \ldots \circ S_n \sqsubseteq R$ with $n > 1$, $R$ is non-simple
- for any subproperty axiom $S \sqsubseteq R$ with $S$ non-simple, $R$ is non-simple
- all other properties are simple

Example: $Q \circ P \sqsubseteq R \quad R \circ P \sqsubseteq R \quad R \sqsubseteq S \quad P \sqsubseteq R \quad Q \sqsubseteq S$

non-simple: $R, S$  

simple: $P, Q$
Property Characteristics

- a property can be
  - the inverse of another property: hasParent $\equiv$ parentOf $^{-}$
    
    rule version:
    
    hasParent(x,y) $\rightarrow$ parentOf(y,x)
    
    parentOf(x,y) $\rightarrow$ hasParent(y,x)

- disjoint with another property: Disj(hasParent,parentOf)
  
  rule version:
  
  hasParent(x,y), parentOf(x,y) $\rightarrow$

- other property characteristics that can be expressed:
  (inverse) functionality, transitivity, symmetry, asymmetry, reflexivity, irreflexivity
OWL 2 DL – Semantics

- model-theoretic semantics
- starts with interpretations
- an interpretation maps

individual names, class names and property names...

...into a domain
OWL 2 DL – Semantics

- mapping is extended to complex class expressions:
  - $T^I = \Delta^I$
  - $\bot^I = \emptyset$
  - $(C \cap D)^I = C^I \cap D^I$
  - $(C \cup D)^I = C^I \cup D^I$
  - $(\neg C)^I = \Delta^I \setminus C^I$
  - $\forall R.C = \{ x \mid \forall y. (x,y) \in R^I \to y \in C^I \}$
  - $\exists R.C = \{ x \mid \exists y. (x,y) \in R^I \land y \in C^I \}$
  - $\geq n R.C = \{ x \mid \# \{ y \mid (x,y) \in R^I \land y \in C^I \} \geq n \}$
  - $\leq n R.C = \{ x \mid \# \{ y \mid (x,y) \in R^I \land y \in C^I \} \leq n \}$

- ...and to role expressions:
  - $U^I = \Delta^I \times \Delta^I$
  - $(R^{-})^I = \{ (y,x) \mid (x,y) \in R^I \}$

- ...and to axioms:
  - $C(a)^I$ holds, if $a^I \in C^I$
  - $R(a,b)^I$ holds, if $(a^I, b^I) \in R^I$
  - $C \sqsubseteq D^I$ holds, if $C^I \subseteq D^I$
  - $R \sqsubseteq S^I$ holds, if $R^I \subseteq S^I$
  - $\text{Disj}(R, S)^I$ holds if $R^I \cap S^I = \emptyset$
  - $S_1 \circ S_2 \circ ... \circ S_n \sqsubseteq R^I$ holds if $S_1^I \circ S_2^I \circ ... \circ S_n^I \subseteq R^I$
but often OWL 2 DL is said to be a fragment of FOL...

yes, there is a translation of OWL 2 DL into FOL

...which (interpreted under FOL semantics) coincides with the definition just given.
OWL in Practice: Tools

- **Editors** ([http://semanticweb.org/wiki/Editors](http://semanticweb.org/wiki/Editors))
  - Most common editor: [Protégé 4](http://semanticweb.org/wiki/Editors)
  - Other tools: [TopBraid Composer ($)](http://semanticweb.org/wiki/Editors), [NeOn toolkit](http://semanticweb.org/wiki/Editors)
  - Special purpose apps, esp. for light-weight ontologies (e.g. [FOAF editors](http://semanticweb.org/wiki/Editors))

  - OWL DL: [Pellet](http://semanticweb.org/wiki/Reasoners), [HermiT](http://semanticweb.org/wiki/Reasoners), [FaCT++]($), [RacerPro ($)](http://semanticweb.org/wiki/Reasoners)
  - OWL EL: [CEL](http://semanticweb.org/wiki/Reasoners), [SHER](http://semanticweb.org/wiki/Reasoners), [snorocket ($)](http://semanticweb.org/wiki/Reasoners), [ELLY](http://semanticweb.org/wiki/Reasoners) (extension of [IRIS](http://semanticweb.org/wiki/Reasoners))
  - OWL RL: [OWLIM](http://semanticweb.org/wiki/Reasoners), [Jena](http://semanticweb.org/wiki/Reasoners), [Oracle Prime](http://semanticweb.org/wiki/Reasoners) (part of O 11g) ($),
  - OWL QL: [Owlgres](http://semanticweb.org/wiki/Reasoners), [QuOnto](http://semanticweb.org/wiki/Reasoners), [Quill](http://semanticweb.org/wiki/Reasoners)

- Many tools use the **OWL API** library (Java)

- Note: many other **Semantic Web tools** are found online
Summary

- OWL
  - ontology language based on description logics
  - basic axioms: class and property inclusions
  - construction of complex class descriptions from simple ones
  - model-theoretic “extensional“ semantics
  - alternative: translation to FOL
  - tool support for editing and automated reasoning available

- Next: Coffee Break
THANK YOU.

Sebastian Rudolph and Barış Sertkaya
FCA for OWL Ontologies

Sebastian Rudolph\textsuperscript{1}  Barış Sertkaya\textsuperscript{2}

\textsuperscript{1}Institut AIFB  
Karlsruhe Institute of Technology  
Germany

\textsuperscript{2}SAP Research Center Dresden  
Germany

Enhancing OWL Ontologies via Formal Concept Analysis  
Tutorial @ IJCAI 2011
Outline

1. Motivation
2. Ontology Completion
3. FCA with Open World Semantics
4. Implications in Partial Contexts
5. Attribute Exploration in Partial Contexts
6. Ontology Completion
7. Tool Demonstration
Ontologies are formal representations of application domains

provide vocabulary for representing knowledge, and methods for reasoning with this knowledge

building blocks of the Semantic Web, successful applications in medicine, biology, and bioinformatics

written in OWL, W3C standard

tool support needed for increasing number of ontologies with large sizes!
Motivation

Existing tools support:

1. detecting inconsistencies
2. inferring consequences
3. finding reasons for them

What about completeness?

- are there missing relations between classes?
- are there missing individuals?
- if so, how to extend the ontology appropriately?
Motivation

Existing tools support:

1. detecting inconsistencies
2. inferring consequences
3. finding reasons for them

What about completeness?

- are there
  - missing relations between classes?
  - missing individuals?
- if so, how to extend the ontology appropriately?
Motivation

Existing tools support:

1. detecting inconsistencies
2. inferring consequences
3. finding reasons for them

What about completeness?

- are there
  - missing relations between classes?
  - missing individuals?
- if so, how to extend the ontology appropriately?
Motivation

Existing tools support:

1. detecting inconsistencies
2. inferring consequences
3. finding reasons for them

What about completeness?

- are there
  - missing relations between classes?
  - missing individuals?
- if so, how to extend the ontology appropriately?
Motivation

Existing tools support:

1. detecting inconsistencies
2. inferring consequences
3. finding reasons for them

What about completeness?

- are there
  - missing relations between classes?
  - missing individuals?
- if so, how to extend the ontology appropriately?
Ontology Completion

\[ \text{EUmember} \equiv \text{Country} \sqcap \exists \text{isMemberOf.}\{\text{EU}\} \]
\[ \text{Mediterranean} \equiv \text{Country} \sqcap \exists \text{hasBorderTo.}\{\text{MediterraneanSea}\} \]
\[ \text{Asian} \equiv \text{Country} \sqcap \exists \text{hasTerritoryIn.}\{\text{Asia}\} \]
\[ \text{European} \equiv \ldots \]

- All European countries are EU members?
- All EU members that have a border to the Mediterranean Sea have territories in Europe?
The Phosphatases Ontology

OWL Ontology for human protein phosphatase family

[Wolstencroft et al. (2005) Wolstencroft, Brass, Horrocks, Lord, Sattler, Turi, & Stevens]

- developed based on peer-reviewed publications
- detailed knowledge about different classes of such proteins
- **TBox**: classes of proteins, relations among these classes
- **ABox**: large set of human phosphatases identified and documented by expert biologists

Given this ontology the biologist wants to know:

1. Are there relations that hold in the real world, but that do not follow from the TBox?
2. Are there phosphatases that are not represented in the ABox, or even that have not yet been identified?
The Phosphatases Ontology

OWL Ontology for human protein phosphatase family


- developed based on peer-reviewed publications
- detailed knowledge about different classes of such proteins
- **TBox**: classes of proteins, relations among these classes
- **ABox**: large set of human phosphatases identified and documented by expert biologists

Given this ontology the biologist wants to know:

1. Are there relations that hold in the real world, but that do not follow from the TBox?
2. Are there phosphatases that are not represented in the ABox, or even that have not yet been identified?
missing relations in the TBox?
missing individuals in the ABox?

cannot be answered by an automated tool alone

a domain expert is required!

capture expert knowledge with the minimum number of questions

avoid questions that can be answered via DL reasoning

may require literature, new experiments, thus discovery of new knowledge!
FCA comes into play

- field of mathematics for **conceptual data analysis**
- derive a classification from a given set of samples

**Attribute exploration**: knowledge acquisition method of FCA
- asks successive questions to a **domain expert**
- triggers discovery of missing relations and objects
- the resultant knowledge is **complete** w.r.t. the application domain
  - guarantees the **minimum** number of questions
- earliest application in lattice theory itself led to discovery of new theorem in the field [Reeg & Weiβ(1990)]
but the problem is ...

- formal contexts based on closed world assumption
- DL knowledge bases based on open world assumption!

- in DLs \((\mathcal{T}, \mathcal{A}) \not\models C(i)\) does not mean \((\mathcal{T}, \mathcal{A}) \models \neg C(i)\)!
- in FCA a cross in row \(g\), column \(m\) means \(g\) has attribute \(m\), and no cross means \(g\) does not have attribute \(m\)
- previous work on extending FCA to open-world semantics [Ganter(1999), Burmeister & Holzer(2000), Holzer(2004a), Holzer(2004b)]
- designed for a usage scenario different from ours
- the resulting knowledge still incomplete: implications valid in some completions, implications valid in all completions
- we want complete knowledge, descriptions of provided individuals are allowed to be incomplete
but the problem is ...

- formal contexts based on closed world assumption
- DL knowledge bases based on open world assumption!

- in DLs \((\mathcal{T}, \mathcal{A}) \not\models C(i)\) does not mean \((\mathcal{T}, \mathcal{A}) \models \neg C(i)\)!
- in FCA a cross in row \(g\), column \(m\) means \(g\) has attribute \(m\), and no cross means \(g\) does not have attribute \(m\)

- previous work on extending FCA to open-world semantics [Ganter(1999), Burmeister & Holzer(2000), Holzer(2004a), Holzer(2004b)]
- designed for a usage scenario different from ours
- the resulting knowledge still incomplete: implications valid in some completions, implications valid in all completions
- we want complete knowledge, descriptions of provided individuals are allowed to be incomplete
but the problem is ...

- formal contexts based on closed world assumption
- DL knowledge bases based on open world assumption!

- in DLs $(\mathcal{T}, \mathcal{A}) \not\models C(i)$ does not mean $(\mathcal{T}, \mathcal{A}) \models \neg C(i)$!
- in FCA a cross in row $g$, column $m$ means $g$ has attribute $m$, and no cross means $g$ does not have attribute $m$
- previous work on extending FCA to open-world semantics [Ganter(1999), Burmeister & Holzer(2000), Holzer(2004a), Holzer(2004b)]
- designed for a usage scenario different from ours
- the resulting knowledge still incomplete: implications valid in some completions, implications valid in all completions
- we want complete knowledge, descriptions of provided individuals are allowed to be incomplete
but the problem is ...

- formal contexts based on closed world assumption
- DL knowledge bases based on open world assumption!

- in DLs \((\mathcal{T}, \mathcal{A}) \not\models C(i)\) does not mean \((\mathcal{T}, \mathcal{A}) \models \neg C(i)\)!
- in FCA a cross in row \(g\), column \(m\) means \(g\) has attribute \(m\), and no cross means \(g\) does not have attribute \(m\)
- previous work on extending FCA to open-world semantics [Ganter(1999), Burmeister & Holzer(2000), Holzer(2004a), Holzer(2004b)]
- designed for a usage scenario different from ours
- the resulting knowledge still incomplete: implications valid in some completions, implications valid in all completions
- we want complete knowledge, descriptions of provided individuals are allowed to be incomplete
Partial object description (pod):
a tuple \((A, S)\) where \(A, S \subseteq M\) and \(A \cap S = \emptyset\).

called a full object description if \(A \cup S = M\)

partial context \(\mathcal{K}\): a set of pods

full context \(\overline{\mathcal{K}}\): a set of fods

(a pod \((A', S')\) extends if \(A \subseteq A'\) and \(S \subseteq S'\) (written \((A, S) \leq (A', S')\))

partial context \(\mathcal{K}'\) extends \(\mathcal{K}\) if every pod in \(\mathcal{K}\) is extended by a pod in \(\mathcal{K}'\) (written \(\mathcal{K} \leq \mathcal{K}'\))

full context \(\overline{\mathcal{K}}\) is a realizer of \(\mathcal{K}\) if \(\mathcal{K} \leq \overline{\mathcal{K}}\)

Implications in Partial Contexts

$L \rightarrow R$ is refuted by the pod $(A, S)$ if $L \subseteq A$ and $R \cap S \neq \emptyset$

$L \rightarrow R$ is refuted by $\mathcal{K}$ if it contains a refuting pod

- $Imp(\mathcal{K})$: the set of implications that are not refuted by $\mathcal{K}$
- $Mod(\mathcal{L})$: the set of fods that do not refute $\mathcal{L}$
Attribute Exploration in Partial Contexts

Scenario:
- The expert has access to full context $\overline{\mathcal{K}}$
- Can answer implication questions w.r.t. $\overline{\mathcal{K}}$
- Although it might need new experiments, etc.
- Counterexamples are allowed to be partial

Formally:
- Initial (possibly empty) partial context $\mathcal{K}$
- Initially empty implications $\mathcal{L}$
- Full context $\overline{\mathcal{K}}$, realizer of $\mathcal{K}$
- Expert answers questions $L \rightarrow R$ w.r.t. $\overline{\mathcal{K}}$
  - If yes, then $L \rightarrow R$ not refuted by $\overline{\mathcal{K}}$, add to $\mathcal{L}$
  - If no, then extend $\mathcal{K}$ s.t. it refutes $L \rightarrow R$
Consequently, the following invariant will be satisfied:

$$K \leq \overline{K} \subseteq \text{Mod}(\mathcal{L})$$

Our aim is to enrich $\mathcal{K}$ and $\mathcal{L}$ s.t.:

1. $\mathcal{L}$ is not only sound, but also complete for $\text{Imp}(\overline{\mathcal{K}})$
2. $\mathcal{K}$ refutes all other implications (refuted by $\overline{\mathcal{K}}$)

An implication undecided w.r.t. $\mathcal{K}$ and $\mathcal{L}$ if it does not follow from $\mathcal{L}$ and is not refuted by $\mathcal{K}$

find and decide undecided implications s.t. 1 and 2 hold

ask as few questions as possible!
how to find the undecided implications?

- as in the classical case, the premises of the implications are pseudo-intents
- enumerate them in a particular lexicographic order to guarantee minimum number of questions
- ask the expert whether the implication holds
- if yes
  - extend $\mathcal{L}$ with the new implication
  - for $(A, S) \in \mathcal{K}$ close $A$ under $\mathcal{L}$
(\(\mathcal{T}, \mathcal{A}\)) is **complete** w.r.t. the application domain \(\mathcal{I}\) if the following are equivalent: (\(L\) and \(R\) are sets of concept names)

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

- \(2^n \times 2^n\) questions (\(n\) the number of concept names)
- many of them redundant
- do not bother the domain expert unnecessarily
- a smart way to organize these questions!
When is an ontology (formally) complete?

\((\mathcal{T}, \mathcal{A})\) is complete w.r.t. the application domain \(\mathcal{I}\) if the following are equivalent: (\(L\) and \(R\) are sets of concept names)

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

- \(2^n \times 2^n\) questions (\(n\) the number of concept names)
- many of them redundant
- do not bother the domain expert unnecessarily
- a smart way to organize these questions!
Ontology Completion

- extended attribute exploration to open world semantics of DL ontologies
- proved termination, correction, and minimum number of questions
- integrated a DL reasoner for avoiding questions that can be answered by DL reasoning
- improved usability:
  - detecting errors
  - recovery from errors
  - deferring questions
Ontology Completion process

When a question is asked:
- first check if it follows from the ontology
- if not ask the expert
- if the expert confirms, add a new axiom to the ontology
- if the expert has a counterexample, add it to the ontology

Is it true that instances of C1, C2 are also instances of C3, C4?

Does not follow from the TBox and ABox

NO! I know a counterexample. (The ABox is extended with the counterexample)

YES! It is true. (The TBox is updated with the new implication)
OntoComP Protégé 4 Plugin

http://ontocomp.googlecode.com
Ontologies inevitable in many applications: Semantic web, medicine,...

increasing need for tool support for maintaining quality

completeness, a quality dimension that has not yet been considered

novel knowledge acquisition algorithm from FCA for ontology completion

implementation as Protégé 4 plugin

- OntoComp http://ontocomp.googlecode.com
- OntoComp library http://ontocomplib.googlecode.com
- FCA library http://fcalib.googlecode.com


