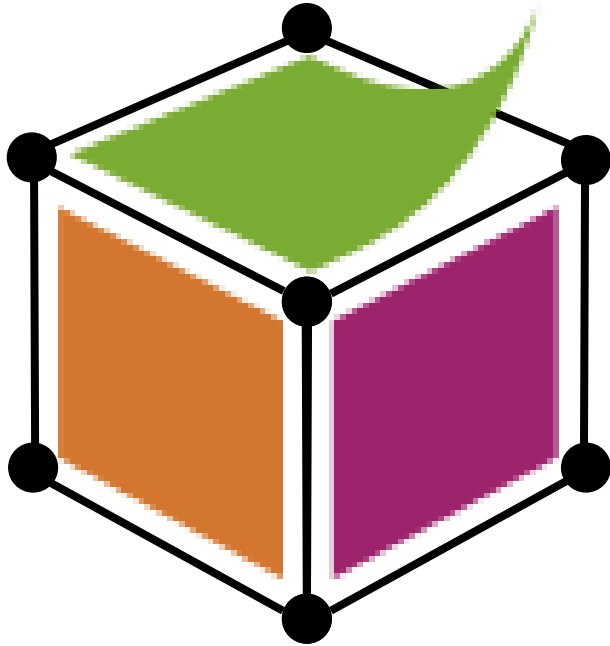


Enhancing OWL Ontologies via Formal Concept Analysis

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



INTRODUCTION

Sebastian Rudolph and Barış Sertkaya

This Tutorial – Presenters

2

□ Sebastian Rudolph



- Institut AIFB, Karlsruhe Institute of Technology
- PhD in Mathematics (TU Dresden)
- <http://www.sebastian-rudolph.de>

□ Barış Sertkaya



- SAP Research Center Dresden
- PhD in Computer Science (TU Dresden)
- <http://sites.google.com/site/sertkayabaris>

Tutorial Homepage:

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

This Tutorial – Schedule

3

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

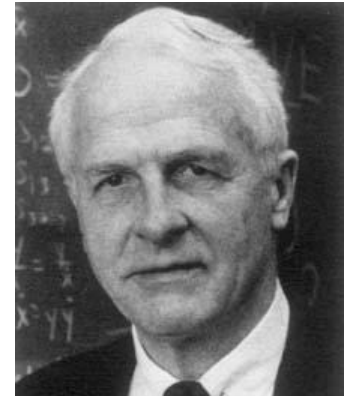
17:45-18:30 Software demonstration
and hands-on session



Formal Concept Analysis?

4

- 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?

5

- 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

6

□ 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)**

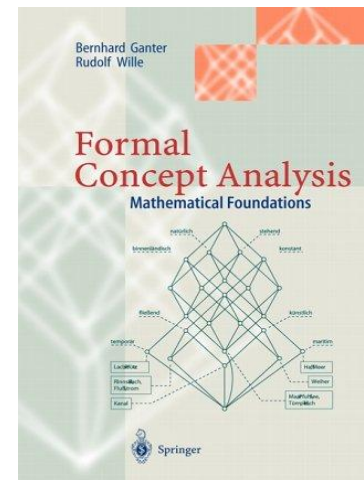
▣ next: October 2011, Nancy, France

□ Monograph

▣ Bernhard Ganter & Rudolf Wille.

„Formal Concept Analysis. Mathematical Foundations“

Springer Verlag, 1999

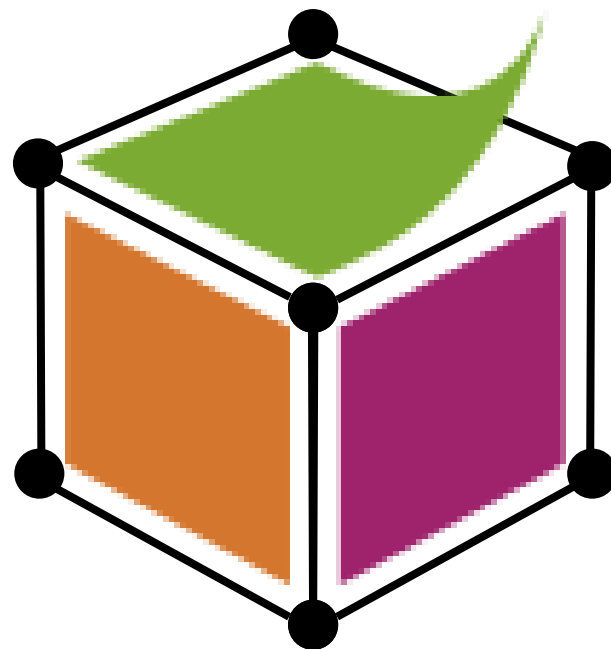


□ FCA Website by Uta Priss: <http://www.upriss.org.uk/fca/fca.html>

Acknowledgements

7

- 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/>



FOUNDATIONS OF FORMAL CONCEPT ANALYSIS

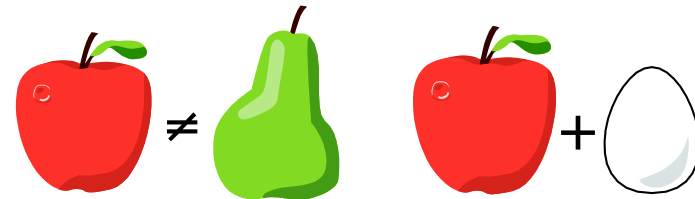
16.07.2011

Sebastian Rudolph and Barış Sertkaya

Introduction

2

- 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?

3

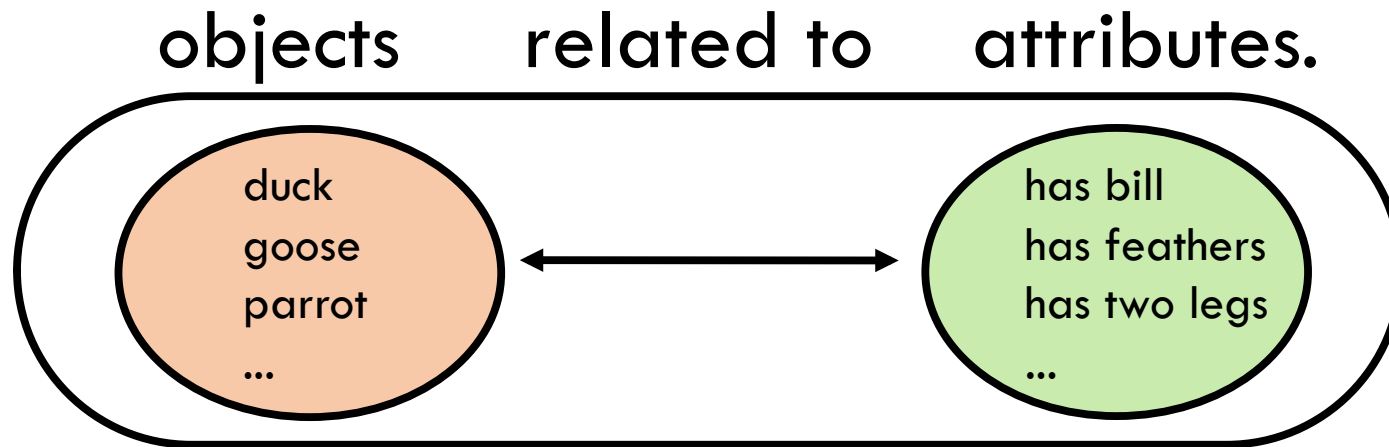
- Consider the concept “bird”. What drives us to call something a “bird” ?
- Every object with certain **attributes** is called “bird”:
 - ◆ A bird has feathers.
 - ◆ A bird has two legs.
 - ◆ A bird has a bill. ...
- All **objects** having these attributes are called “birds”:
 - ◆ Duck, goose, owl and parrot are birds.
 - ◆ Penguins are birds, too.
 - ◆ ...



What is a Concept?

4

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

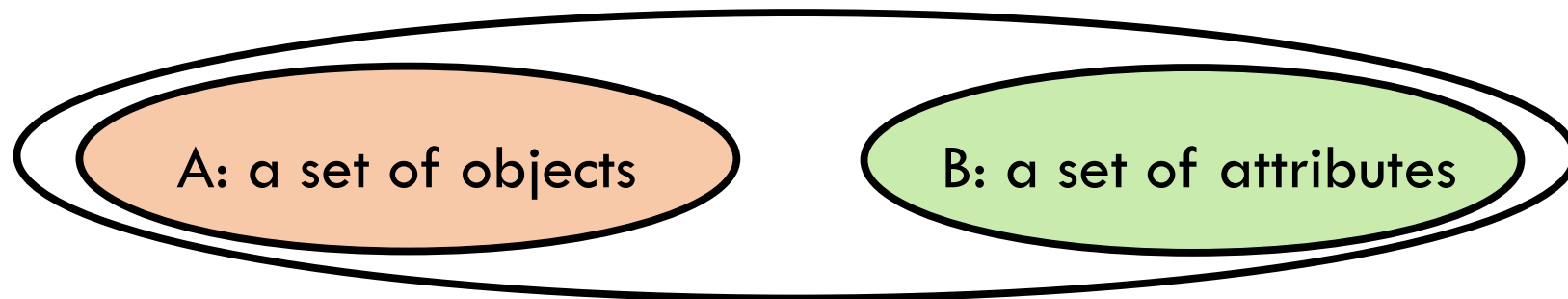


➔ Objects, attributes and a relation form a **formal concept**.

What is a Concept?

5

- So, a **formal concept** is constituted by two parts

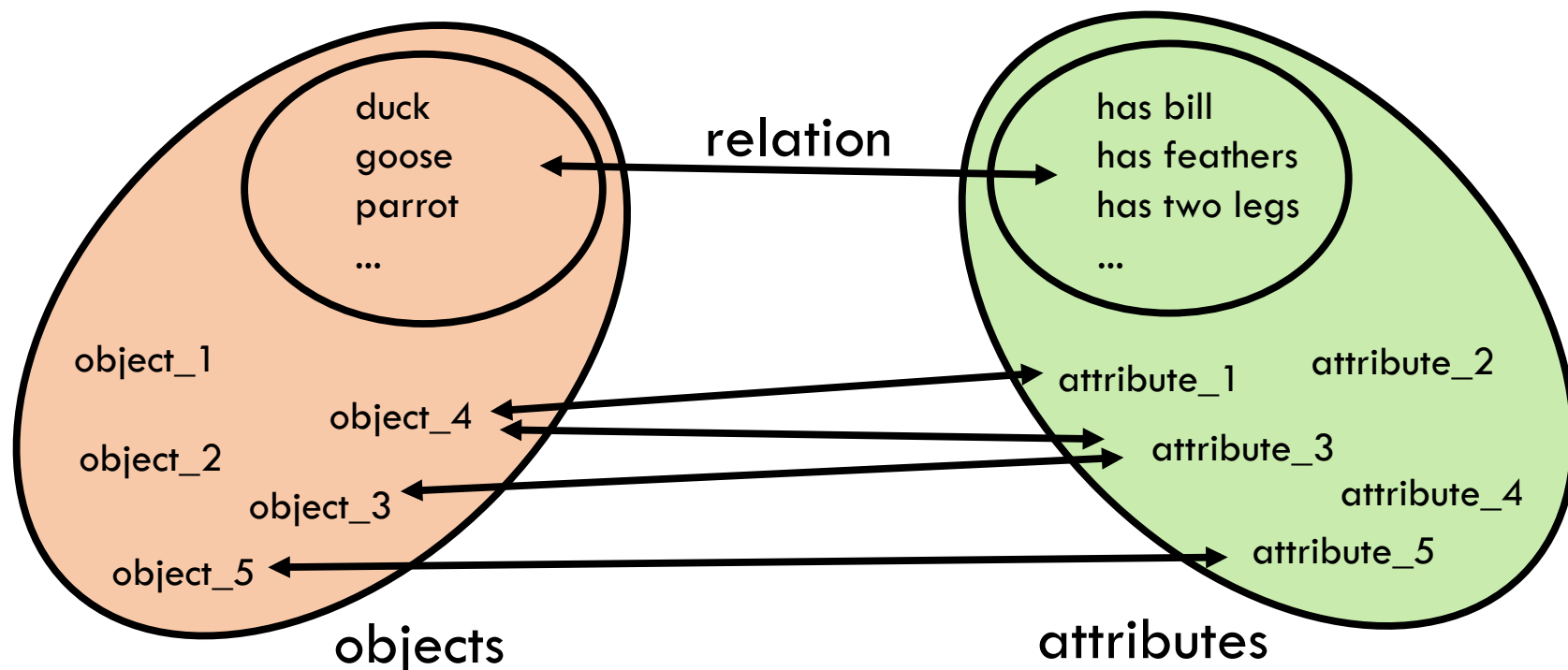


- ... having a certain relation:
 - every **object** belonging to this concept has all the attributes in B
 - every **attribute** belonging to this concept is shared by all objects in A
- **A** is called the **concept's extent**, **B** is called the concept's **intent**

The Universe of Discourse

6

- A repertoire of **objects** and **attributes** (which might or might not be related) constitutes the „context“ of our considerations.



The Formal Context

K	small	medium	big	2legs	4legs	feathers	hair	fly	hunt	run	swim	mane	hooves
dove	x			x		x		x					
hen	x			x		x							
duck	x			x		x		x			x		
goose	x			x									
owl	x			x									
hawk	x			x									
eagle		x		x		x		x	x				
fox		x			x		x		x	x			
dog		x			x								
wolf		x			x								
cat	x				x								
tiger			x		x		x		x				
lion			x		x		x		x	x			
horse			x		x		x			x			
zebra					x		x			x			
cow					x		x						

set of **attributes** (M)

crosses indicate **incidence relation** (I) between G and M

(G,M,I) is called **formal context**

set of **objects** (G)



Definition of Formal Concepts

8

- 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

9

- 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.

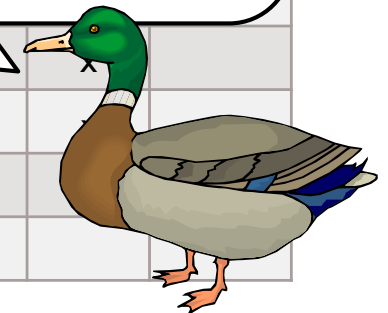
Calculating Formal Concepts

10

K	small	medium	big	2legs	4legs	feathers	hair	fly	hunt	run	swim	mane	hooves
dove	x			x		x		x					
hen	x			x		x							
duck	x			x		x		x			x		
goose	x			x		x		x			x		
owl	x			x		x		x	x				
hawk	x			x		x		x	x				

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: $(A'', A') = (\{\text{duck, goose}\}, \{\text{small, 2legs, feathers, fly, swim}\})$

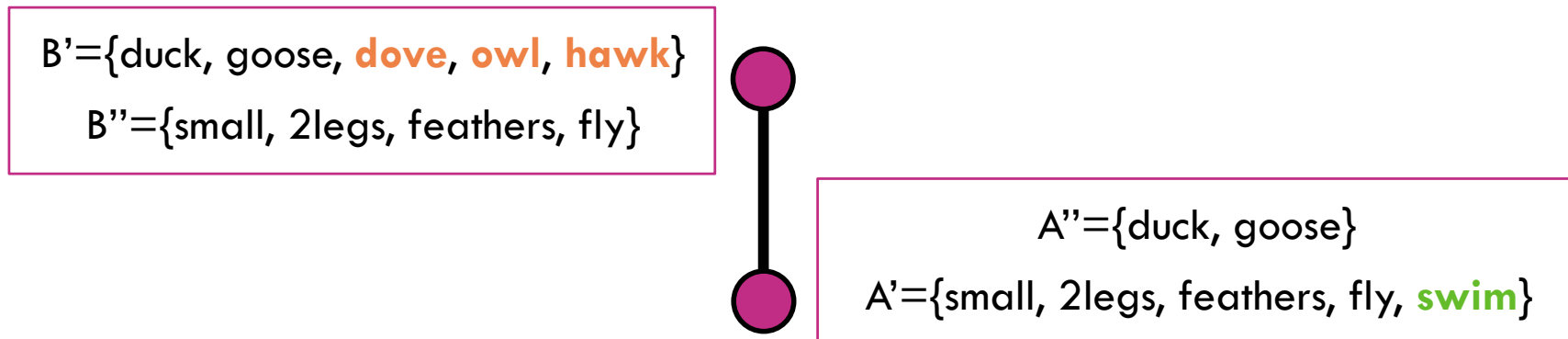
lion			x		x		x		x	x			
horse			x		x		x			x			
zebra			x		x		x			x			
cow			x		x		x						



Ordering Concepts

11

The **formal concept** (A'', A') ($A'' = \{\text{duck, goose}\}$, $A' = \{\text{small, 2legs, feathers, fly, swim}\}$) is represented in the line diagram as a node:



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

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

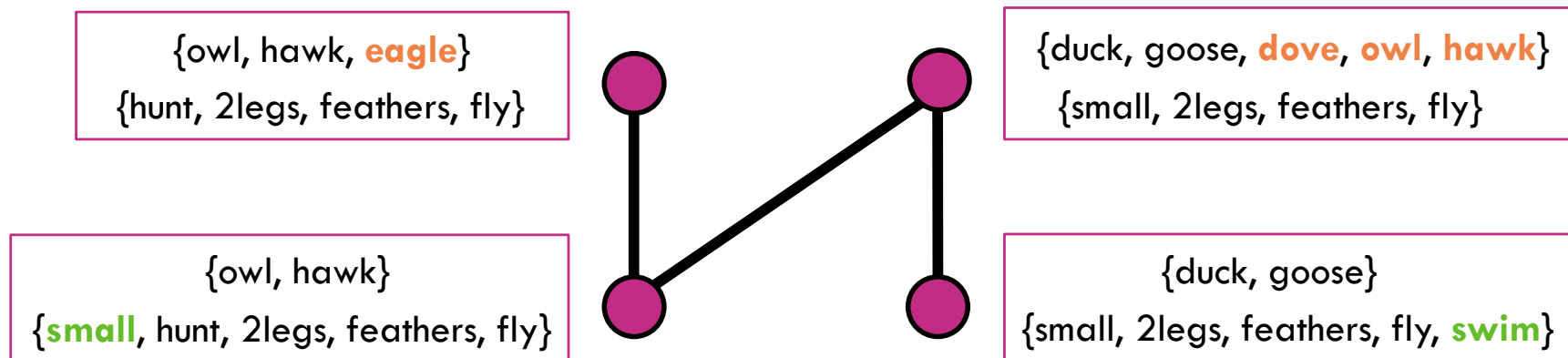
12

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**:

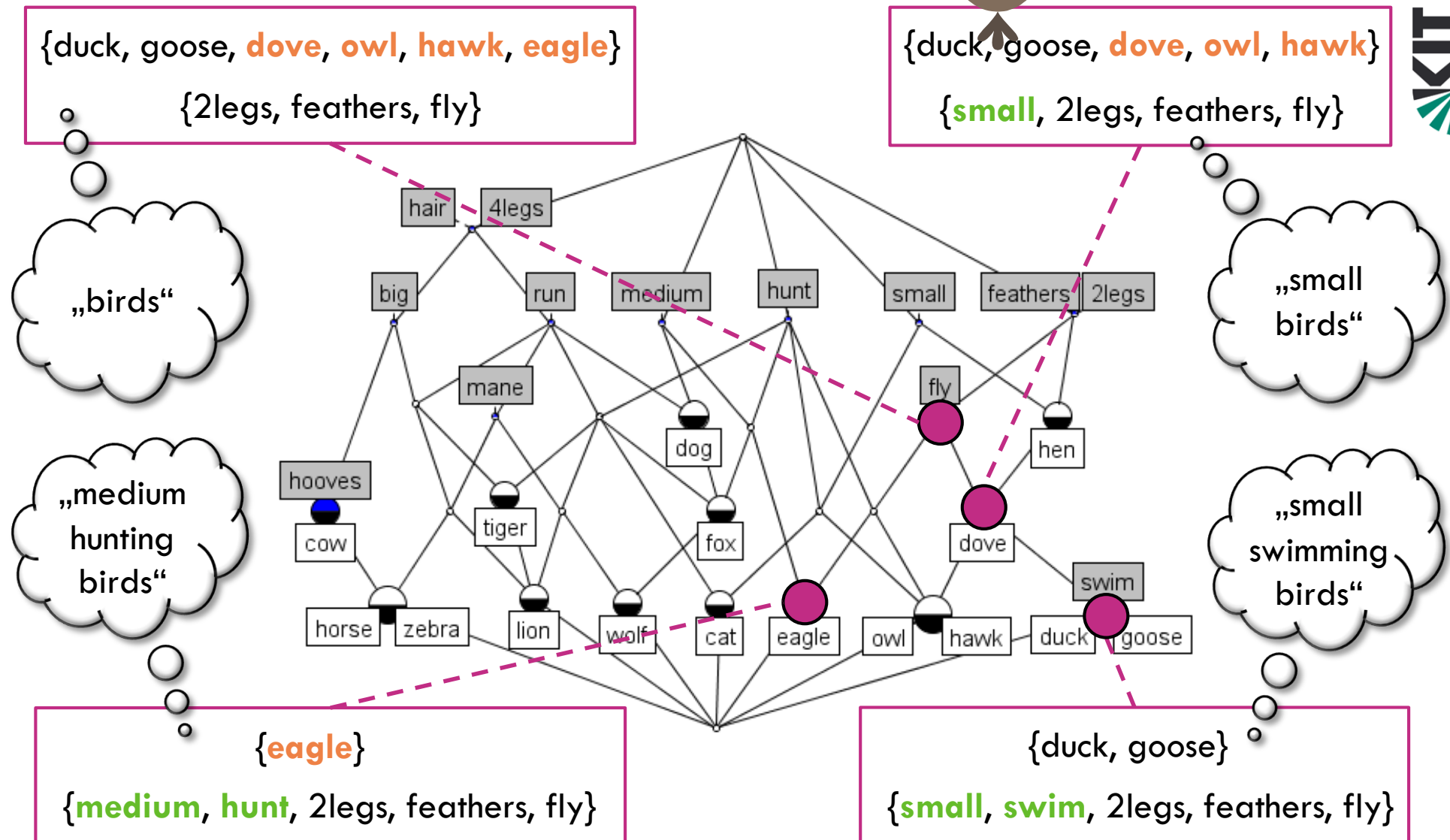
Cut over extents, Ganter's algorithm etc.

The Concept Lattice

- The subconcept–superconcept relation defines an order $\leq$ 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:
$$(A_1, A_2) \leq (B_1, B_2) \Leftrightarrow A_1 \subseteq B_1 (\Leftrightarrow B_2 \subseteq A_2)$$
- (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, $(B, \leq)$ 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 $\underline{B} = (B, \leq)$ plus the last property forms a mathematical structure: the **concept lattice**.

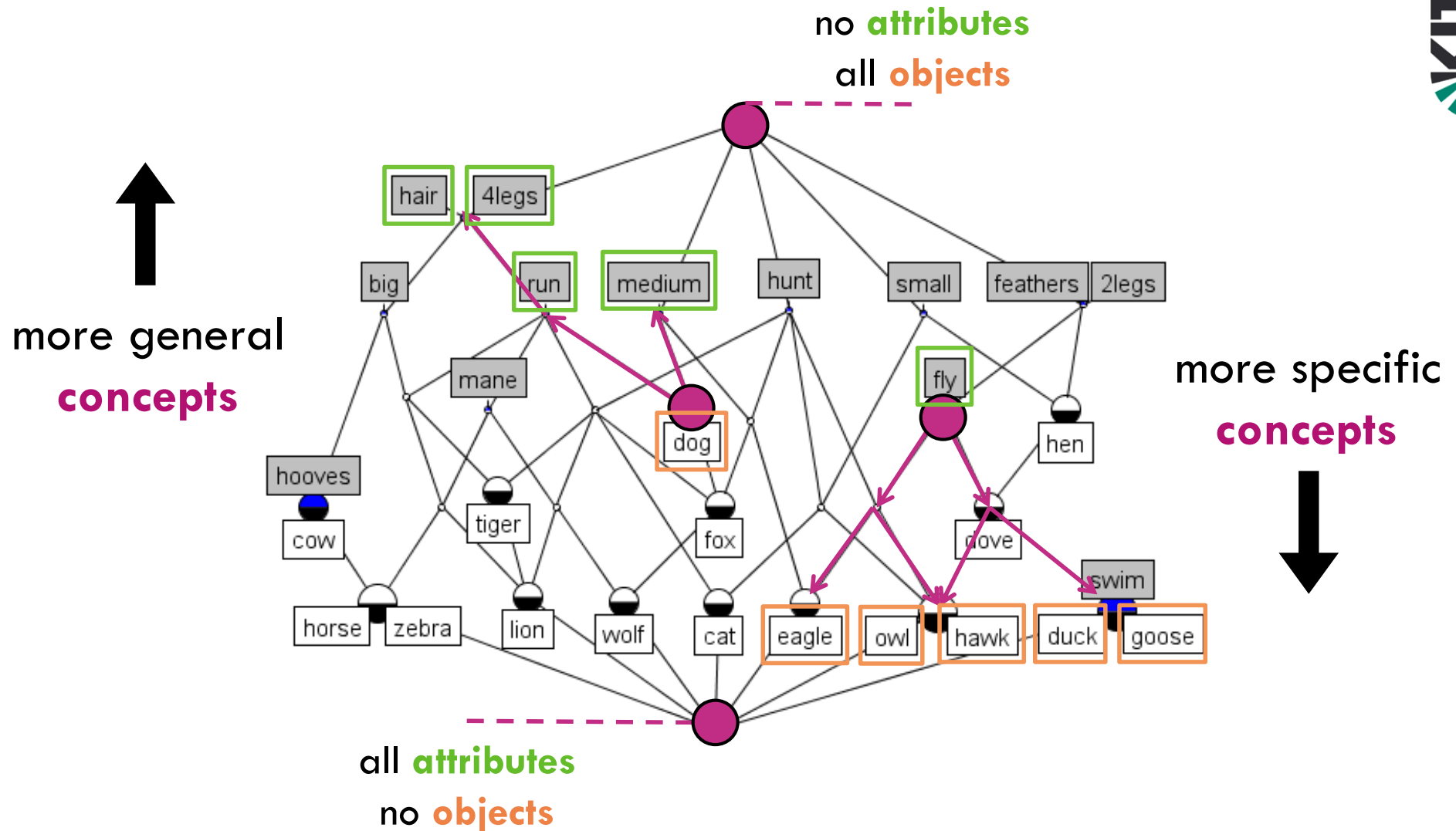
Concept Lattice – Formal Concepts

14

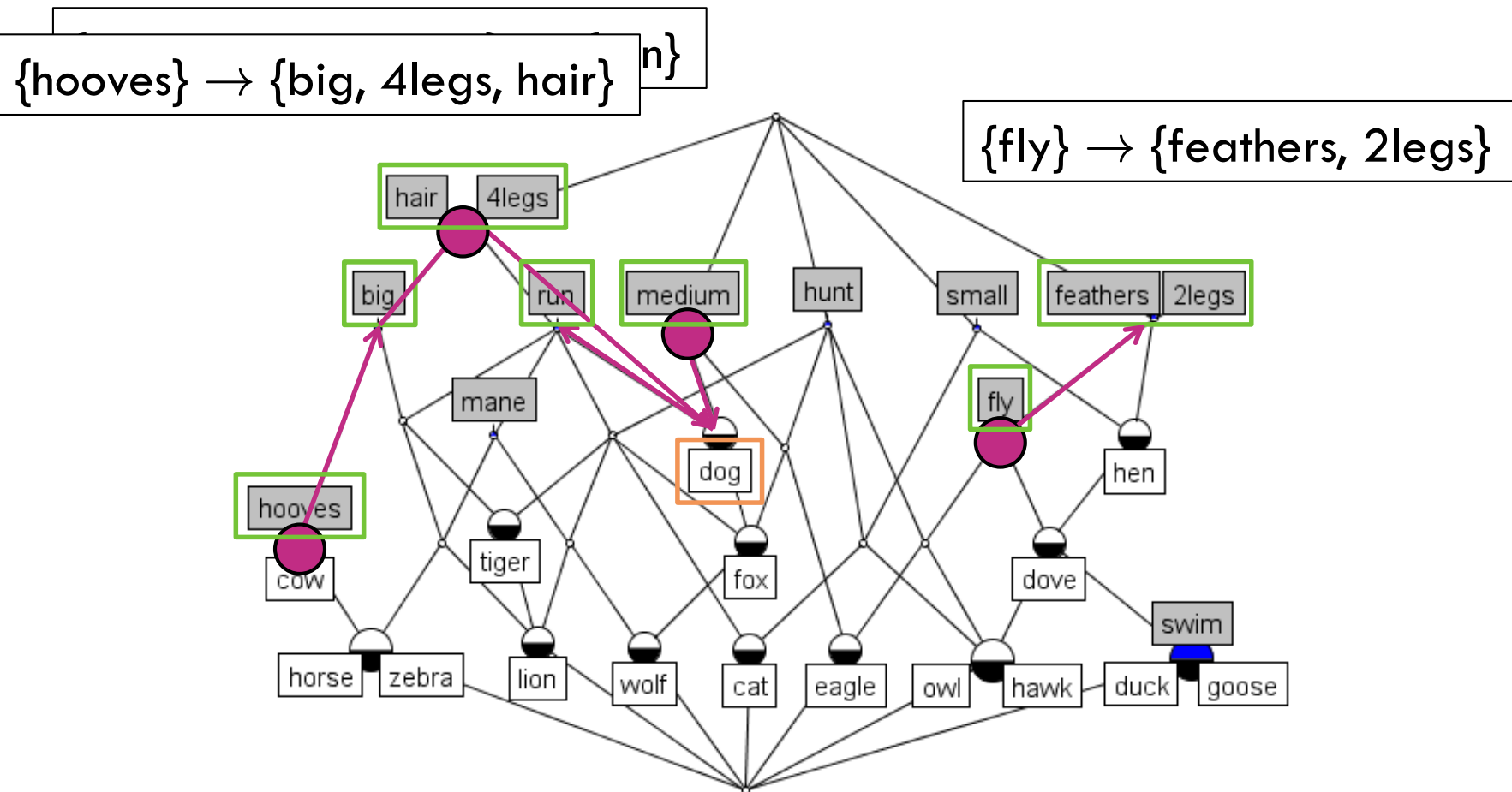


Concept Lattice – Top and Bottom

15



Concept Lattice – Implications



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

Summary

17

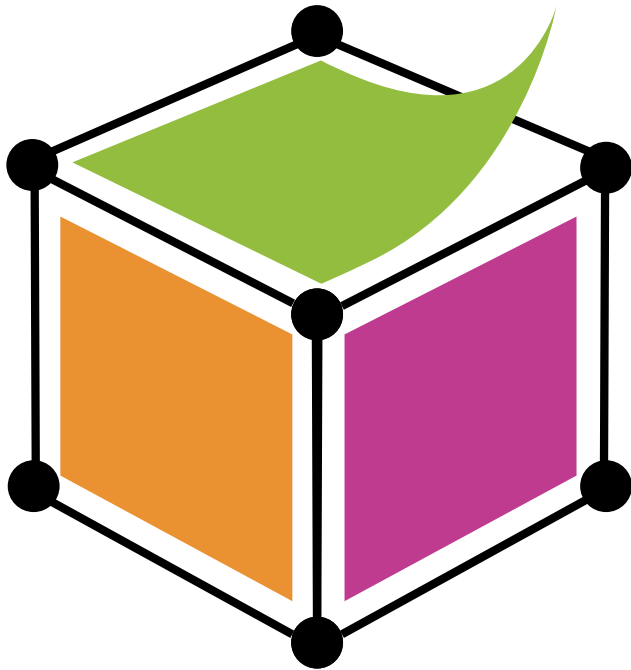
- Formal contexts
 - **Objects**, **attributes** and **incidence relation**
- ... and **formal concepts**
 - Extent and intent
 - Subconcept relations
- Concept lattices
 - How to interpret a concept lattice
 - Generalization and specialization
 - Implications
- Next: **attribute exploration**

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)

2

- For $A, B \subseteq M$, the *implication* $A \rightarrow B$ holds in $\mathbb{K}$, 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:

- $\{\text{wet}\} \rightarrow \{\text{fluid}\}$
- $\{\text{fluid, dry}\} \rightarrow \{\text{warm}\}$
- $\{\text{dry, wet}\} \rightarrow \{\text{cold}\}$ (!)

$\mathbb{K}$	fluid	dry	wet	warm	cold
Earth		x			x
Water	x		x		x
Air	x		x	x	
Fire	x	x		x	

How to „Datamine“ Implications?

3

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

K	fluid	dry	wet	warm	cold
Earth		x			x
Water	x		x		x
Air	x		x	x	
Fire	x	x		x	

How to „Datamine“ Implications?

4

- 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

5

- Given a formal context $\mathbb{K}$, a set of implications $\mathfrak{S}$ is called *implication base* of $\mathbb{K}$, if ...
 - ▣ every implication $A \rightarrow B$ from $\mathfrak{S}$ holds in $\mathbb{K}$,
 - ▣ every implication $A \rightarrow B$ holding in $\mathbb{K}$ can be derived from $\mathfrak{S}$, and
 - ▣ none of the implications from $\mathfrak{S}$ can be derived from the other implications contained in $\mathfrak{S}$
- Question: which $A \rightarrow A''$ to choose to make up an implication base?

The Stem Base

6

- Question: which $A \rightarrow A''$ to choose to make up an implication base?
- Answer: take all the *pseudo-intents* of $\mathbb{K}$.
- 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

7

- 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 $\mathfrak{S}(A)$ for the result of applying implication set $\mathfrak{S}$ 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

8

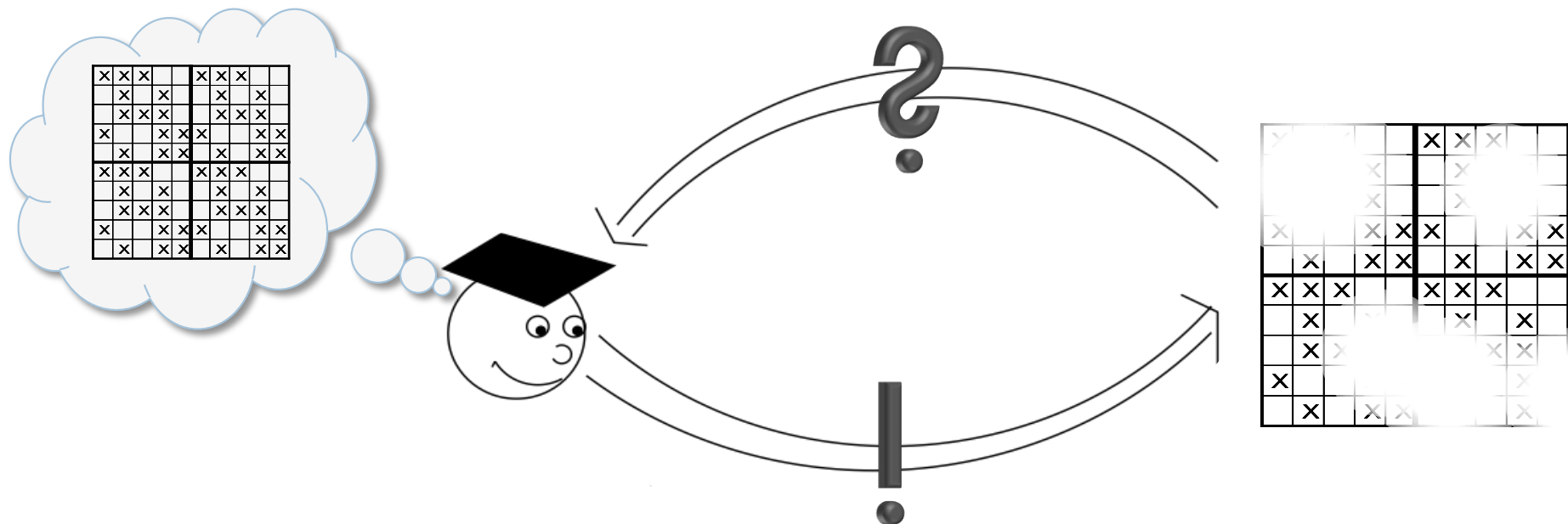
- Input formal context $\mathbb{K}$
- Create list $\mathfrak{S}$ of implications, initially empty
Let $A = [0,0,\dots,0]$ (*bit representation of empty set*)
- Repeat
 - ▣ Add $A \rightarrow A''$ to $\mathfrak{S}$ in case $A \neq A''$
 - ▣ Starting from $i = |M| + 1$, decrement i until
 - $i=0$ or
 -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

... **i** ...

A: [0,0,1,0,1,1,0]
 A+i: [0,0,1,1,0,0,0]
 $\mathfrak{S}(A+i)$: [0,0,1,1,0,1,1]

Interactive Knowledge Acquisition via Attribute Exploration

- Sometimes, $\mathbb{K}$ is not entirely known from the beginning, but implicitly present as an expert's knowledge
- Attribute exploration determines the stembase of $\mathbb{K}$ by asking expert for missing information



Interactive Knowledge Acquisition via Attribute Exploration

10

- Sometimes, $\mathbb{K}$ is not entirely known from the beginning, but implicitly present as an expert's knowledge
- Attribute exploration determines the stembase of $\mathbb{K}$ 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

11

- Input formal context $\underline{\mathbb{K}} = (H, M, J)$ where $J = (H \times M) \cap I$
- Create list $\mathfrak{S}$ of implications, initially empty
Let $A = [0, 0, \dots, 0]$ (*bit representation of empty set*)
- Repeat
 - ▣ Add $A \rightarrow A''$ to $\mathfrak{S}$ 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 $\mathfrak{S}$ to $A + i$ produces 1s only at i
 - ▣ If $i = 0$ output $\mathfrak{S}$ and exit
 - ▣ Let $A = \mathfrak{S}(A + i)$

Has to be altered, because implication valid in $\underline{\mathbb{K}}$ might be invalid in $\mathbb{K}$ since refuted by an object not yet recorded. Then augmenting $\underline{\mathbb{K}}$ by this object allows to refine the hypothesis.

Making It Interactive...

12

- Instead of just adding $A \rightarrow A''$ to $\mathfrak{S}$, do the following control Loop:
 - While $A \neq A''$
 - Ask expert whether $A \rightarrow A''$ is valid in $\mathbb{K}$
 - If yes, add $A \rightarrow A''$ to $\mathfrak{S}$ and exit while-loop, otherwise ask for counterexample and add it to $\underline{\mathbb{K}}$
- 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 $\underline{\mathbb{K}}=(H,M,J)$?
 - $H_{\text{new}} = H \cup \{g\}$
 - $J_{\text{new}} = J \cup \{(g,m) \mid m \text{ is attribute of } g \text{ in } \mathbb{K}\}$
 - Essentially: just add a line to the cross table

Making It Interactive...

13

- Instead of just adding $A \rightarrow A''$ to $\mathfrak{S}$, do the following control loop:
 - While $A \neq A'$,
 - Ask expert whether $A \rightarrow A''$ is valid in $\mathbb{K}$
 - If yes, add $A \rightarrow A''$ to $\mathfrak{S}$ and exit while-loop, otherwise ask for counterexample g and add it to $\underline{\mathbb{K}}$
- Remarks:
 - Attribute set of g has to comply with the implications already confirmed
 - Changing $\underline{\mathbb{K}}$ 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

14

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

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* (Zickwolff 1991)
 - ▣ DLs with bounded role depth: *relational exploration* (Rudolph 2004)

THANK YOU.

Sebastian Rudolph and Barış Sertkaya

References I

- [Burmeister, Holzer 2005] Burmeister, P., Holzer, R., Treating Incomplete Knowledge in Formal Concept Analysis, In: B. Ganter, G. Stumme, R. Wille (Eds.): Formal Concept Analysis: State of the Art. LNAI 3626. Springer, Heidelberg 2005.
- [Ganter 1987] Ganter, B.: Algorithmen zur formalen Begriffsanalyse. In: B. Ganter, R. Wille, K.E. Wolf (Eds.): Beiträge zur Begriffsanalyse, pages 241– 254, B.I. Wissenschaftsverlag, Mannheim, 1987.
- [Ganter 1996] Ganter, B.: Attribute exploration with background knowledge. In: Theoretical Computer Science 217(2), pages 215– 233, 1999.
- [Ganter 1984] Ganter, B.: Two Basic Algorithms in Concept Analysis. FB4-Preprint 831, TH Darmstadt, 1984.
- [Ganter & Wille 1999] Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, 1999.

References II

18

- [Guigues & Duquenne 1986] Guigues, J.-L., Duquenne, V.: Familles minimales d'implications informatives resultant d'un tableau de données binaires. In: Math. Sci Humaines 95, pages 5–18, 1986.
- [Rudolph 2004] Rudolph, S.: Exploring Relational Structures via FLE . In: K.E. Wolff, H.D. Pfeiffer, H.S. Delugach (Eds.): Conceptual Structures at Work, ICCS 2004, pages 196–212, Huntsville, USA, LNCS 3127, Springer, 2004.
- [Stumme 1996] Stumme, G.: Attribute Exploration with Background Implications and Exceptions, In: H.-H. Bock, W. Polasek (eds.): Data analysis and information systems, Springer, 457--469, 1996.
- [Zickwolff 1991] Zickwolff, M.: Rule Exploration: First Order Logic in Formal Concept Analysis. PhD thesis, TH Darmstadt, 1991.

Enhancing OWL Ontologies via Formal Concept Analysis

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



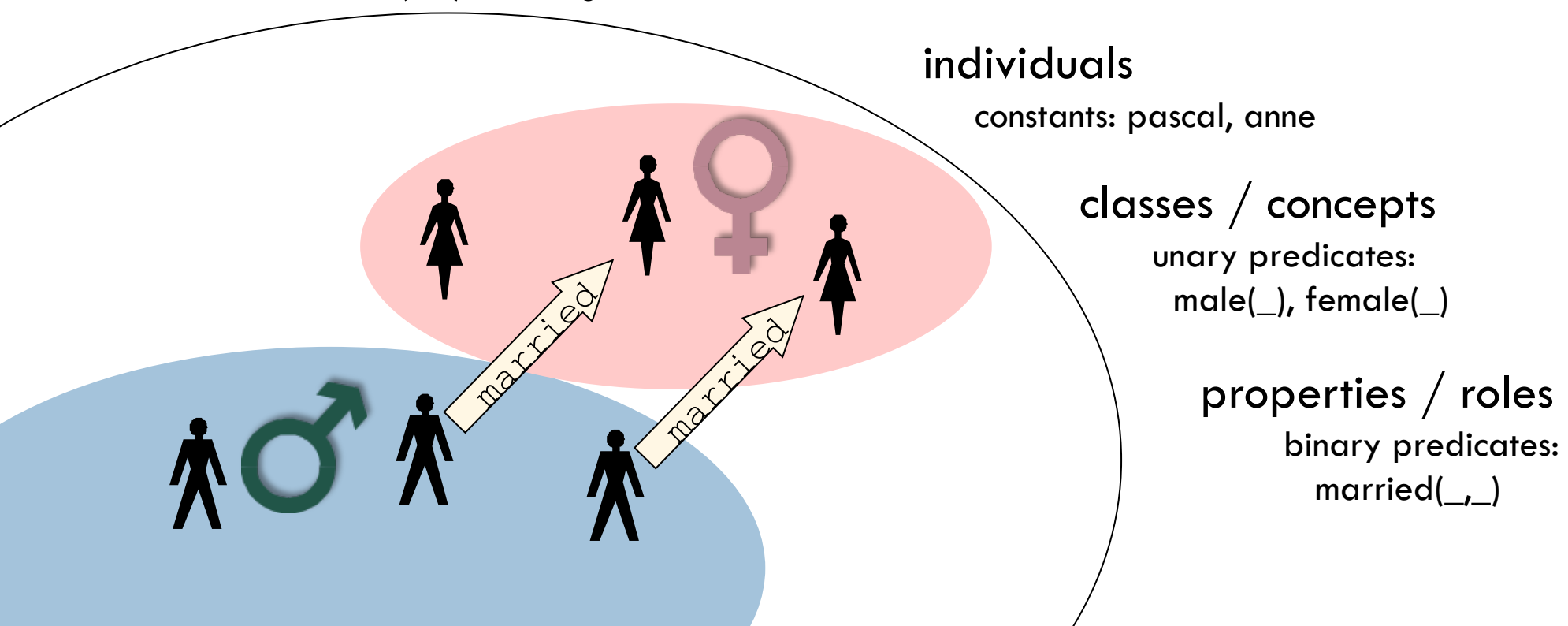
FOUNDATIONS OF OWL

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



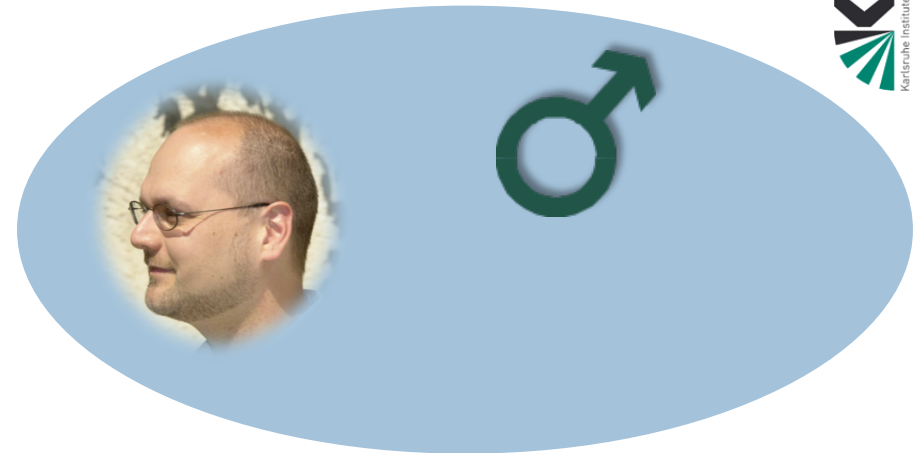
Assertional Knowledge

- asserts information about concrete named individuals

- class membership: Male(Pascal)

`<Male rdf:about="pascal"/>`

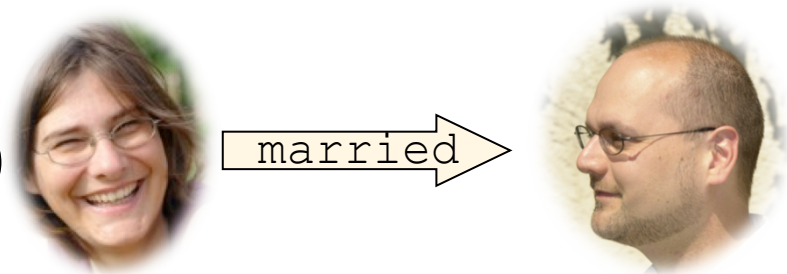
rule version: `→ Male(pascal)`



- property membership: married(anne,pascal)

`<rdf:Description rdf:about="anne">
 <married rdf:resource="pascal"/>
</rdf:Description>`

rule version: `→ married (anne,pascal)`



That's all what can be said with RDF!

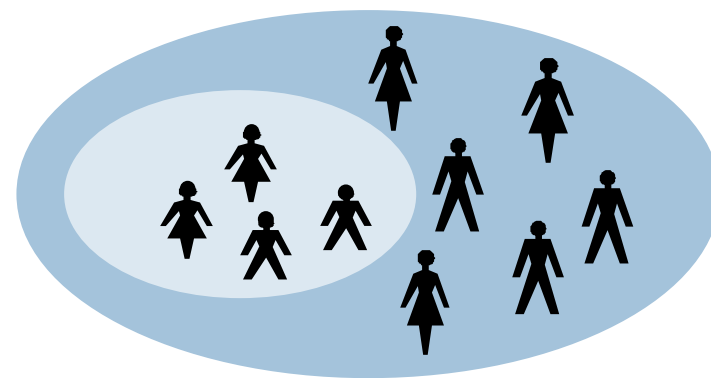
Terminological Knowledge – Subclasses and Subproperties

- information about how classes and properties relate in general

- subclass: $\text{Child} \sqsubseteq \text{Person}$

```
<owl:Class rdf:about="Woman">  
  <rdfs:subClassOf rdf:resource="Person"/>  
</owl:Class>
```

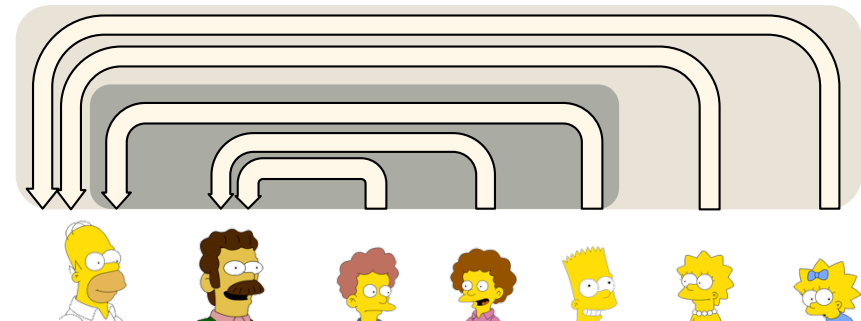
rule version: $\text{Child}(x) \rightarrow \text{Person}(x)$



- subproperty: $\text{sonOf} \sqsubseteq \text{childOf}$

```
<owl:ObjectProperty rdf:about="sonOf">  
  <rdfs:subPropertyOf rdf:resource="childOf"/>  
</owl:ObjectProperty>
```

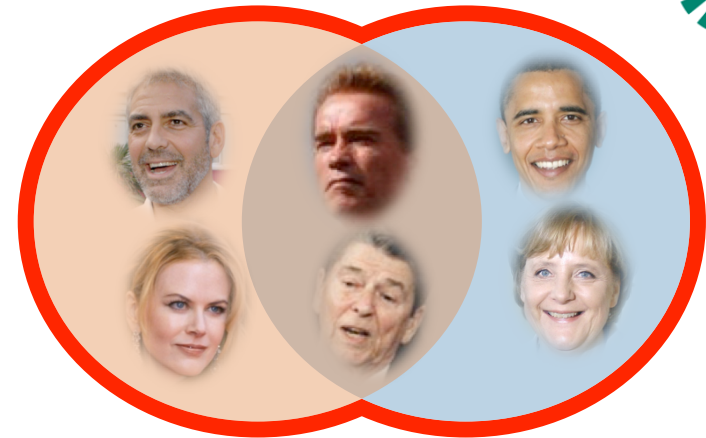
rule version: $\text{sonOf}(x,y) \rightarrow \text{childOf}(x,y)$



Class Constructors

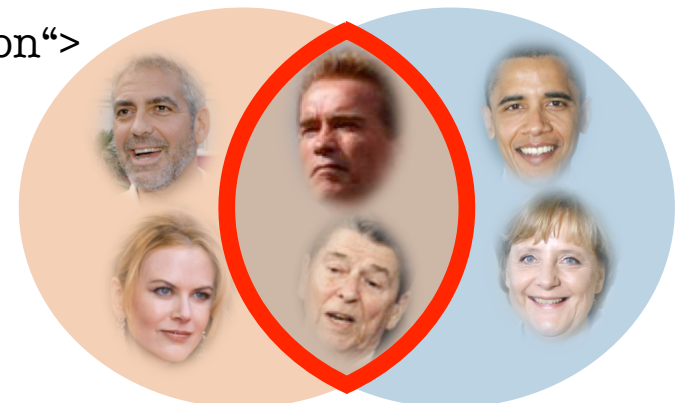
- build new classes from class, property and individual names
 - ▣ union: Actor $\sqcup$ Politician

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



- ▣ intersection: Actor $\sqcap$ Politician

```
<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: $\neg$ Politician

```
<owl:complementOf  
  rdf:resource="Politician">
```



- closed classes: {anne,merula,pascal}

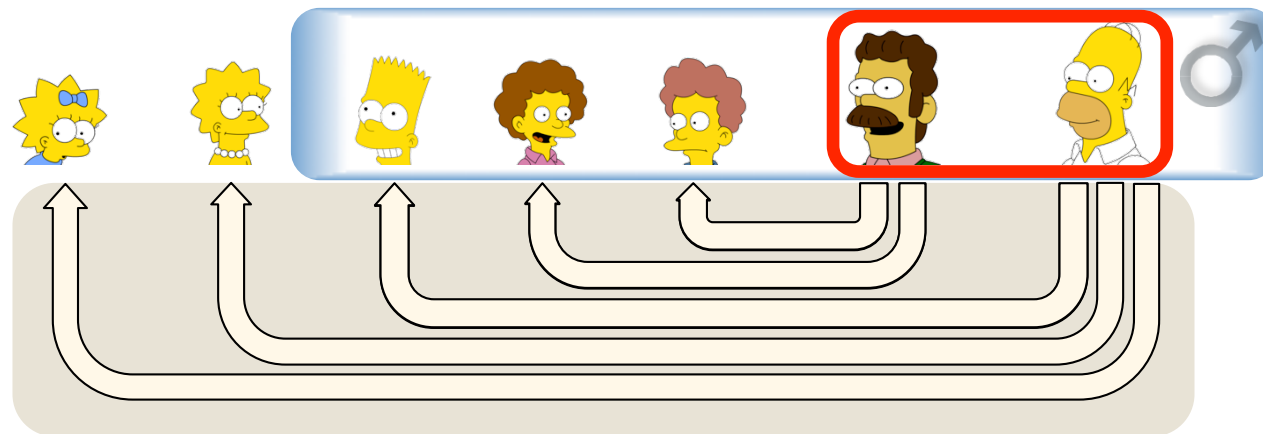
```
<owl:oneOf rdf:parseType="Collection">  
  <rdf:Description rdf:about="anne"/>  
  <rdf:Description rdf:about="merula"/>  
  <rdf:Description rdf:about="pascal"/>  
</owl:oneOf>
```



Class Constructors

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

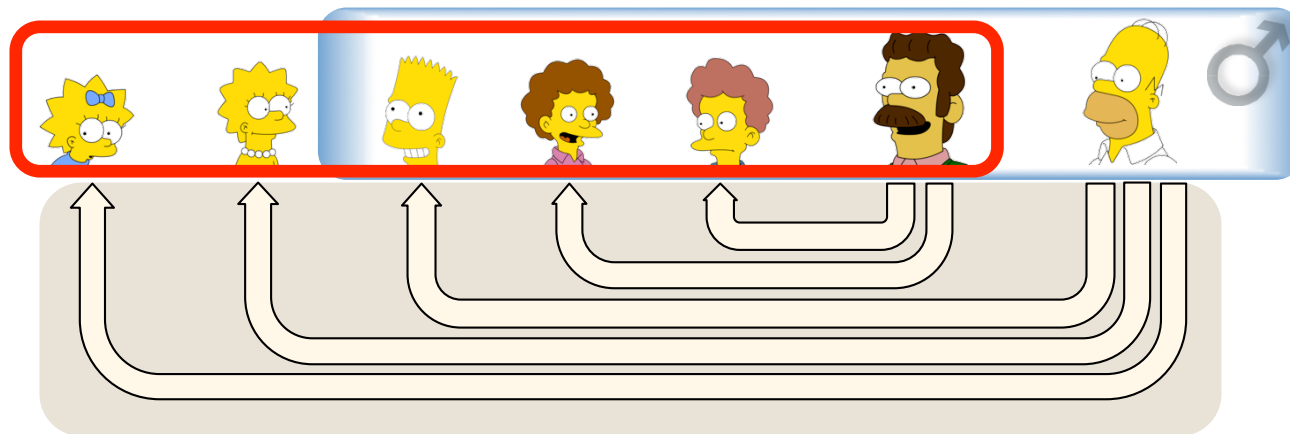
```
<owl:Restriction>  
  <owl:onProperty rdf:resource="hasChild"/>  
  <owl:someValuesFrom rdf:resource="Male"/>  
</owl:Restriction>
```



Class Constructors

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

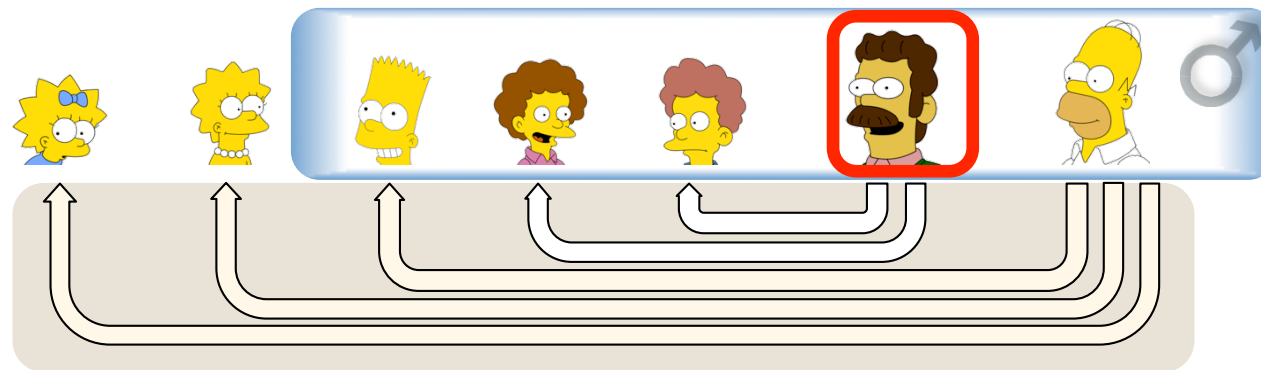
```
<owl:Restriction>  
  <owl:onProperty rdf:resource="hasChild"/>  
  <owl:allValuesFrom rdf:resource="Female"/>  
</owl:Restriction>
```



Class Constructors

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

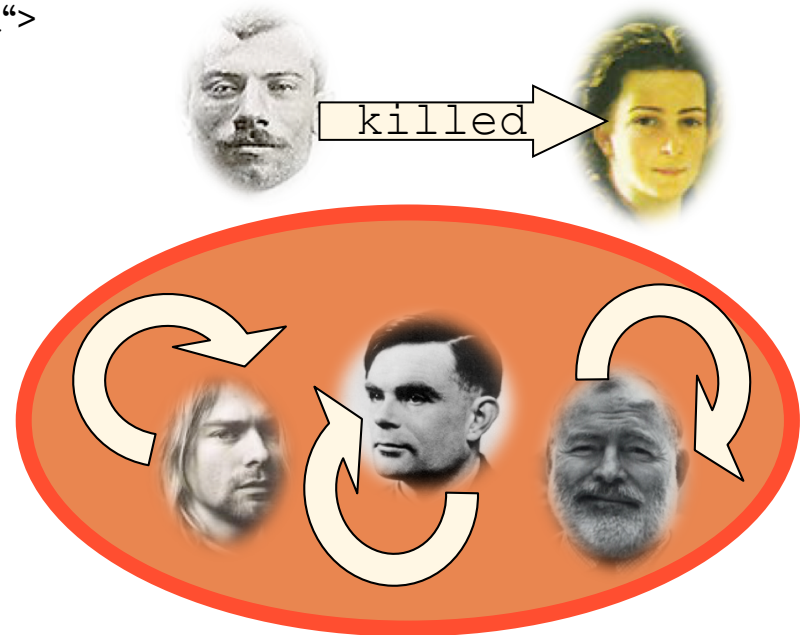
```
<owl:Restriction>  
  <owl:minQualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">  
    2 </owl:minQualifiedCardinality>  
  <owl:onProperty rdf:about="hasChild"/>  
  <owl:onClass rdf:about="Female"/>  
</owl:Restriction>
```



Class Constructors

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

```
<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: $\perp$

- ...“empty” class containing no individuals

- `<owl:Nothing>`

- ▣ universal property: U

- ...property linking every individual to every individual

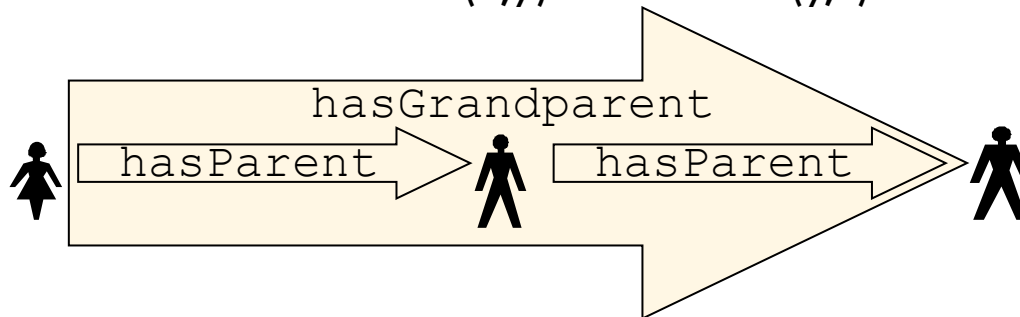
- `<owl:topObjectProperty>`

Property Chain Axioms

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

□ $\text{hasParent} \circ \text{hasParent} \sqsubseteq \text{hasGrandparent}$

rule version: $\text{hasParent}(x,y) \wedge \text{hasParent}(y,z) \rightarrow \text{hasGrandparent}(x,z)$



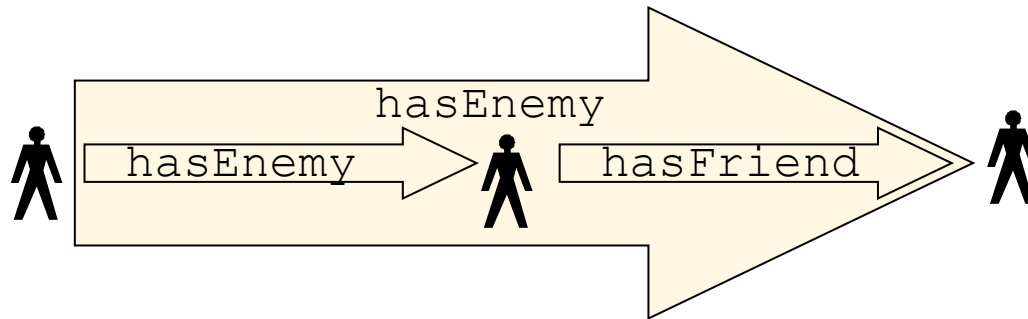
```
<rdf:Description rdf:about="hasGrandparent">
  <owl:propertyChainAxiom rdf:parseType="Collection">
    <owl:ObjectProperty rdf:about="hasParent"/>
    <owl:ObjectProperty rdf:about="hasParent"/>
  </owl:propertyChainAxiom>
</rdf:Description>
```

Property Chain Axioms

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

□ $\text{hasEnemy} \circ \text{hasFriend} \sqsubseteq \text{hasEnemy}$

rule version: $\text{hasEnemy}(x,y) \wedge \text{hasFriend}(y,z) \rightarrow \text{hasEnemy}(x,z)$



```
<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 $<$ on the properties
 - ▣ every property chain axiom has to have one of the following forms:

$R \circ R \sqsubseteq R$	$R^- \sqsubseteq R$	$S_1 \circ S_2 \circ \dots \circ S_n \sqsubseteq R$
$R \circ S_1 \circ S_2 \circ \dots \circ S_n \sqsubseteq R$		$S_1 \circ S_2 \circ \dots \circ S_n \circ R \sqsubseteq R$
 - ▣ thereby, $S_i < R$ for all $i = 1, 2, \dots, n$.
- Example 1: $R \circ S \sqsubseteq R$ $S \circ S \sqsubseteq S$ $R \circ S \circ R \sqsubseteq T$
 $\rightarrow$ regular with order $S < R < T$
- Example 2: $R \circ T \circ S \sqsubseteq T$
 $\rightarrow$ not regular because form not admissible
- Example 3: $R \circ S \sqsubseteq S$ $S \circ R \sqsubseteq R$
 $\rightarrow$ not regular because no adequate order exists

Property Chain Axioms: Caution!

(2/2)

- 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 $S_1 \circ S_2 \circ \dots \circ S_n \sqsubseteq R$ 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: $Q \circ P \sqsubseteq R$ $R \circ P \sqsubseteq R$ $R \sqsubseteq S$ $P \sqsubseteq R$ $Q \sqsubseteq S$
non-simple: R, S simple: P, Q

Property Characteristics

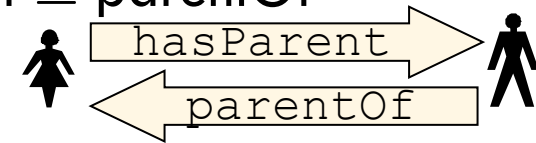
- a property can be

- the inverse of another property: $\text{hasParent} \equiv \text{parentOf}^{-}$

rule version:

$\text{hasParent}(x,y) \rightarrow \text{parentOf}(y,x)$

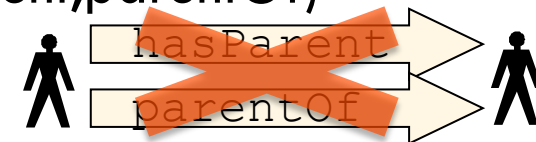
$\text{parentOf}(x,y) \rightarrow \text{hasParent}(y,x)$



- disjoint with another property: $\text{Disj}(\text{hasParent}, \text{parentOf})$

rule version:

$\text{hasParent}(x,y), \text{parentOf}(x,y) \rightarrow$

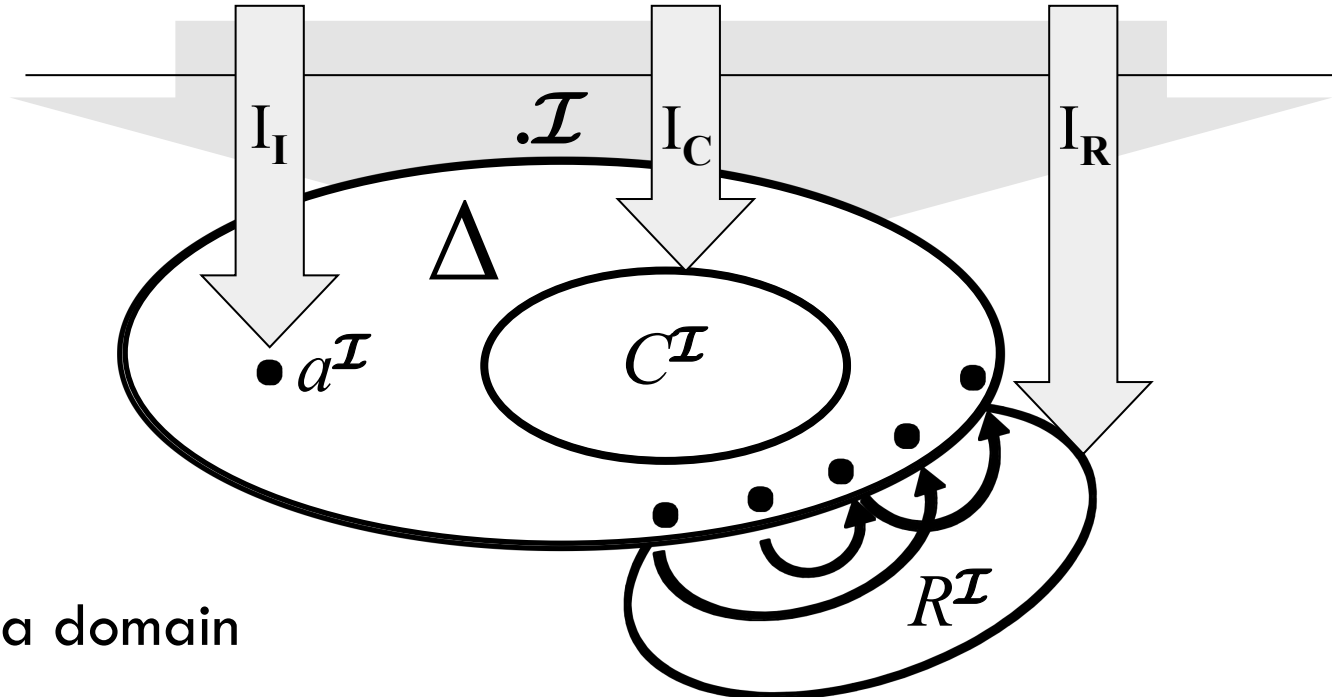


- other property characteristics that can be expressed:
(inverse) functionality, transitivity, symmetry, asymmetry,
reflexivity, irreflexivity

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:

$$\begin{aligned}
 &\square \top^I = \Delta^I & \square \perp^I = \emptyset \\
 &\square (C \sqcap D)^I = C^I \cap D^I & \square (C \sqcup D)^I = C^I \cup D^I & \square (\neg C)^I = \Delta^I \setminus C^I \\
 &\square \forall R.C = \{ x \mid \forall y. (x,y) \in R^I \rightarrow y \in C^I \} \\
 &\square \exists R.C = \{ x \mid \exists y. (x,y) \in R^I \wedge y \in C^I \} \\
 &\square \geq n R.C = \{ x \mid \#\{ y \mid (x,y) \in R^I \wedge y \in C^I \} \geq n \} \\
 &\square \leq n R.C = \{ x \mid \#\{ y \mid (x,y) \in R^I \wedge y \in C^I \} \leq n \}
 \end{aligned}$$

- ...and to role expressions:

$$\square U^I = \Delta^I \times \Delta^I \qquad \square (R^-)^I = \{ (y,x) \mid (x,y) \in R^I \}$$

- ...and to axioms:

$$\begin{aligned}
 &\square C(a) \text{ holds, if } a^I \in C^I & \square R(a,b) \text{ holds, if } (a^I, b^I) \in R^I \\
 &\square C \sqsubseteq D \text{ holds, if } C^I \subseteq D^I & \square R \sqsubseteq S \text{ holds, if } R^I \subseteq S^I \\
 &\square \text{Disj}(R,S) \text{ holds if } R^I \cap S^I = \emptyset \\
 &\square S_1 \circ S_2 \circ \dots \circ S_n \sqsubseteq R \text{ holds if } S_1^I \circ S_2^I \circ \dots \circ S_n^I \subseteq R^I
 \end{aligned}$$

OWL 2 DL – Alternative Semantics

- but often OWL 2 DL is said to be a fragment of FOL...
- yes, there is a translation of OWL 2 DL into FOL

$$\begin{aligned}
 \pi(C \sqsubseteq D) &= (\forall x)(\pi_x(C) \rightarrow \pi_x(D)) \\
 \pi_x(A) &= A(x) \\
 \pi_x(\neg C) &= \neg \pi_x(C) \\
 \pi_x(C \sqcap D) &= \pi_x(C) \wedge \pi_x(D) \\
 \pi_x(C \sqcup D) &= \pi_x(C) \vee \pi_x(D) \\
 \pi_x(\forall R.C) &= (\forall x_1)(R(x, x_1) \rightarrow \pi_{x_1}(C)) \\
 \pi_x(\exists R.C) &= (\exists x_1)(R(x, x_1) \wedge \pi_{x_1}(C)) \\
 \pi_x(\geq n S.C) &= (\exists x_1) \dots (\exists x_n) \left(\bigwedge_{i \neq j} (x_i \neq x_j) \wedge \bigwedge_i (S(x, x_i) \wedge \pi_{x_i}(C)) \right) \\
 \pi_x(\leq n S.C) &= \neg (\exists x_1) \dots (\exists x_{n+1}) \left(\bigwedge_{i \neq j} (x_i \neq x_j) \wedge \bigwedge_i (S(x, x_i) \wedge \pi_{x_i}(C)) \right) \\
 \pi_x(\{a\}) &= (x = a) \\
 \pi_x(\exists S.\text{Self}) &= S(x, x) \\
 \pi(R_1 \sqsubseteq R_2) &= (\forall x)(\forall y)(\pi_{x,y}(R_1) \rightarrow \pi_{x,y}(R_2)) \\
 \pi_{x,y}(S) &= S(x, y) \\
 \pi_{x,y}(R^-) &= \pi_{y,x}(R) \\
 \pi_{x,y}(R_1 \circ \dots \circ R_n) &= (\exists x_1) \dots (\exists x_{n-1}) \\
 &\quad \left(\pi_{x,x_1}(R_1) \wedge \bigwedge_{i=1}^{n-2} \pi_{x_i, x_{i+1}}(R_{i+1}) \wedge \pi_{x_{n-1}, y}(R_n) \right) \\
 \pi(\text{Ref}(R)) &= (\forall x) \pi_{x,x}(R) \\
 \pi(\text{Asy}(R)) &= (\forall x)(\forall y)(\pi_{x,y}(R) \rightarrow \neg \pi_{y,x}(R)) \\
 \pi(\text{Dis}(R_1, R_2)) &= \neg (\exists x)(\exists y)(\pi_{x,y}(R_1) \wedge \pi_{x,y}(R_2))
 \end{aligned}$$

- ...which (interpreted under FOL semantics) coincides with the definition just given.

OWL in Practice: Tools



- Editors (<http://semanticweb.org/wiki/Editors>)
 - Most common editor: [Protégé 4](#)
 - Other tools: [TopBraid Composer](#) (\$), [NeOn toolkit](#)
 - Special purpose apps, esp. for light-weight ontologies (e.g. [FOAF](#) editors)
- Reasoners (<http://semanticweb.org/wiki/Reasoners>)
 - OWL DL: [Pellet](#), [HermiT](#), [FaCT++](#), [RacerPro](#) (\$)
 - OWL EL: [CEL](#), [SHER](#), [snorocket](#) (\$), [ELLY](#) (extension of [IRIS](#))
 - OWL RL: [OwlIM](#), [Jena](#), [Oracle Prime](#) (part of O 11g) (\$),
 - OWL QL: [Owlgres](#), [QuOnto](#), [Quill](#)
- 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 Rudolph¹ Barış Sertkaya²

¹Institut AIFB
Karlsruhe Institute of Technology
Germany

²SAP Research Center Dresden
Germany

Enhancing OWL Ontologies via Formal Concept Analysis
Tutorial @ IJCAI 2011

- 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!



Existing tools
support:

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

quality
dimension of
soundness

What about **completeness**?

- are there
 - **missing relations** between classes?
 - **missing individuals**?
- if so, howto **extend** the ontology appropriately?

Existing tools support:

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

quality
dimension of
soundness

What about completeness?

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

Existing tools
support:

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

quality
dimension of
soundness

What about completeness?

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

Existing tools
support:

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

quality
dimension of
soundness

What about **completeness**?

- are there
 - **missing relations** between classes?
 - **missing individuals**?
- if so, howto **extend** the ontology appropriately?

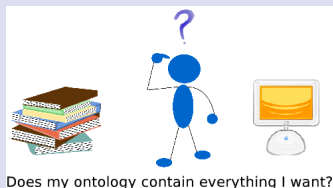
Existing tools
support:

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

quality
dimension of
soundness

What about **completeness**?

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



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



$\mathcal{A}_{\text{countries}}$	Asian	EUmember	European	Mediterranean
Russia	+	?	?	?
China	+	-	-	?
Montenegro	?	?	+	?
Germany	-	+	+	-
Italy	-	+	+	+

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

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?

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?

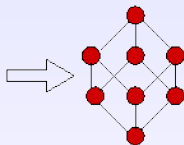
- missing relations in the TBox?
 - missing individuals in the ABox?
- cannot be answered by an automated tool alone
 - a **domain expert** is required!
 - capture expert knowledge with the **minimum number of questions**
 - avoid questions that can be answered via DL reasoning

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

	atr1	atr2	atr3
obj1	x		
obj2	x		x
obj3		x	x
obj4	x	x	



- **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 & Weiß(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 $\mathcal{K} \leq \overline{\mathcal{K}}$

[Baader *et al.*(2007)Baader, Ganter, Sertkaya, & Sattler, Sertkaya(2007)]

$L \rightarrow R$ 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 pods 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 $L \rightarrow R$

Attribute Exploration in Partial Contexts

- 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 $\text{Imp}(\overline{K})$
 - 2 $\mathcal{K}$ refutes all other implications (refuted by $\overline{K}$)

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 $\mathcal{L}$
 -

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 $\Box L \sqsubseteq \Box R$ follows from $\mathcal{T}$
- 3 $(\mathcal{T}, \mathcal{A})$ does not contain a counterexample to $\Box L \sqsubseteq \Box R$

- $2^n \times 2^n$ 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?

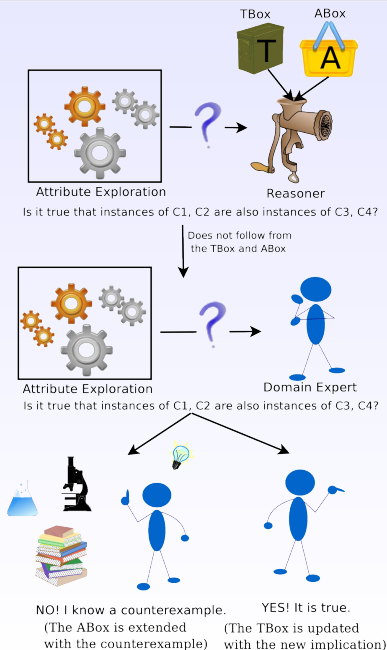
$(\mathcal{T}, \mathcal{A})$ is **complete** w.r.t. the application domain $\mathcal{I}$ if the following are equivalent: (L and R are sets of concept names)

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

- $2^n \times 2^n$ 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!

- 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

ONTOCOMP PROTÉGÉ 4 Plugin

The screenshot displays the ONTOCOMP PROTÉGÉ 4 Plugin interface. On the left, the 'Inferred class hierarchy' shows a tree structure starting with 'Thing', followed by 'Country', which branches into 'Coastal', 'Mediterranean', 'EUmember', and 'GermanSpeaking'. Below 'Country' are 'Ocean', 'Organization', and 'Sea'. The 'GermanSpeaking' class is highlighted. On the right, the 'Ontology Completion' window is active, showing a table with columns for 'Context', 'Coastal', 'Mediterranean', 'EUmember', and 'GermanSpeaking'. The table contains data for Italy, Austria, India, and Germany. Below the table are buttons for 'Start', 'Stop', 'Repair', 'Reset', 'Resume', 'Yes', and 'No'. The 'Messages' pane at the bottom displays a welcome message and a question about instances of 'GermanSpeaking'.

ICFCA09_example.owl (http://www.semanticweb.org/ontologies/2009/0/ICFCA09_example.owl) - [use ↑ - □ X]

File Edit Ontologies Reasoner Tools Refactor Tabs View Window Help

ICFCA09_example.owl

Data Properties Individuals OWLviz DL Query OntoComp Example Tab

Active Ontology Entities Classes Object Properties

Inferred class hierarchy: [Icons]

- Thing
 - Country
 - Coastal
 - Mediterranean
 - EUmember
 - GermanSpeaking
 - Ocean
 - Organization
 - Sea

Ontology Completion: [Icons]

Context Counterexample editor Repair

	Coastal	Mediterranean	EUmember	GermanSpeaking
Italy	+	+	+	-
Austria	-	-	+	?
India	+	-	-	-
Germany	+	-	+	+

Start Stop Repair Reset Resume Yes No

Messages [No, this is not true in my application domain]

Welcome to the **Ontology Completion Plugin**
Drag & drop class names from the class hierarchy into the context, hit start when you are ready

Are instances of **GermanSpeaking**
also instances of **EUmember**, **Coastal**?

- <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>

References



Baader, F., Ganter, B., Sertkaya, B., & Sattler, U. (2007).

Completing description logic knowledge bases using formal concept analysis.

In M. M. Veloso (Ed.), *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI'07)*. AAAI Press, 230–235.



Burmeister, P. & Holzer, R. (2000).

On the treatment of incomplete knowledge in formal concept analysis.

In B. Ganter & G. W. Mineau (Eds.), *Proceedings of the 8th International Conference on Conceptual Structures, (ICCS 2000)*. Springer-Verlag, vol. 1867 of *Lecture Notes in Computer Science*, 385–398.



Ganter, B. (1999).

Attribute exploration with background knowledge.

Theoretical Computer Science, 217(2), 215–233.



Holzer, R. (2004a).

Knowledge acquisition under incomplete knowledge using methods from formal concept analysis: Part I.

Fundamenta Informaticae, 63(1), 17–39.



Holzer, R. (2004b).

Knowledge acquisition under incomplete knowledge using methods from formal concept analysis: Part II.

Fundamenta Informaticae, 63(1), 41–63.



Reeg, S. & Weiß, W. (1990).

Properties of finite lattices.

Masters thesis, TH Darmstadt, Germany.



Sertkaya, B. (2007).

Formal Concept Analysis Methods for Description Logics.

Ph.D. dissertation, Institute of Theoretical Computer Science, TU Dresden, Germany.



Wolstencroft, K., Brass, A., Horrocks, I., Lord, P. W., Sattler, U., Turi, D., & Stevens, R. (2005).

A little semantic web goes a long way in biology.

In *Proceedings of the 4th International Semantic Web Conference, (ISWC 2005)*. Springer-Verlag, vol. 3729 of *Lecture Notes in Computer Science*, 786–800.