

# Vorlesung Semantic Web



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

Vorlesung im Wintersemester 2012/2013

Dr. Heiko Paulheim

Fachgebiet Knowledge Engineering

# Aufgabe 1



- Gegeben folgende Ontologie:

```
:Baby rdfs:subClassOf :Human .  
:Adult rdfs:subClassOf :Human .  
:Andy a :Baby .  
:Andy a :Adult .
```

- Zeigen Sie, dass das Tableau-Verfahren darin keinen Widerspruch erkennt.

# Aufgabe 1



- Dieselbe Ontologie in DL-NNF:
  - Baby  $\sqsubset$  Human
  - Adult  $\sqsubset$  Human
  - Baby(Andy)
  - Adult(Andy)
- Damit können wir jetzt das Tableau-Verfahren starten

# Aufgabe 1

Baby(Andy), Adult(Andy)

Nr	Aussage	Aktion
1	C(a)	Füge C(a) hinzu

# Aufgabe 1



Baby(Andy), Adult(Andy),  
 $(\neg \text{Baby} \sqcup \text{Human})(\text{Andy})$ ,  
 $(\neg \text{Adult} \sqcup \text{Human})(\text{Andy})$

Nr	Aussage	Aktion
3	C	Wähle ein Individuum a, füge C(a) hinzu

# Aufgabe 1



Baby(Andy), Adult(Andy),  
( $\neg$ Adult  $\sqcup$  Human)(Andy),  
 $\neg$ Baby(Andy)

Baby(Andy), Adult(Andy),  
( $\neg$ Adult  $\sqcup$  Human)(Andy),  
Human(Andy).

Nr	Aussage	Aktion
5	$(C \sqcup D)(a)$	Teile das Tableau in T1 und T2. Füge C(a) zu T1, D(a) zu T2 hinzu

# Aufgabe 1



Baby(Andy), Adult(Andy),  
( $\neg$ Adult  $\sqcup$  Human)(Andy),  
 $\neg$ Baby(Andy)

Baby(Andy), Adult(Andy),  
Human(Andy),  
 $\neg$ Adult(Andy)

Baby(Andy), Adult(Andy),  
Human(Andy)

Nr	Aussage	Aktion
5	$(C \sqcup D)(a)$	Teile das Tableau in T1 und T2. Füge C(a) zu T1, D(a) zu T2 hinzu

# Aufgabe 1

- Das dritte Tableau bleibt ohne Widerspruch
  - weitere Axiome können nicht erzeugt werden
- damit erkennt der Reasoner keinen Widerspruch!

# Aufgabe 1

- Fügen Sie nun folgendes zusätzliche Axiom ein:

```
:Baby owl:disjointWith :Adult .
```

- Zeigen Sie, dass das Tableau-Verfahren hier einen Widerspruch finden kann.

# Aufgabe 1

- Damit sind unsere DL-NNF so aus:
  - ¬Baby  $\sqcup$  Human
  - ¬Adult  $\sqcup$  Human
  - ¬Baby  $\sqcup$  ¬Adult
  - Baby(Andy)
  - Adult(Andy)
- ...und das Tableau-Verfahren ist noch nicht fertig

# Aufgabe 1



Baby(Andy), Adult(Andy),  
( $\neg$ Adult  $\sqcup$  Human)(Andy),  
 $\neg$ Baby(Andy)

Baby(Andy), Adult(Andy),  
Human(Andy),  
 $\neg$ Adult(Andy)

Baby(Andy), Adult(Andy),  
Human(Andy),  
( $\neg$ Adult  $\sqcup$   $\neg$ Baby)(Andy)

Nr	Aussage	Aktion
3	C	Wähle ein Individuum a, füge C(a) hinzu

# Aufgabe 1



Baby(Andy), Adult(Andy),  
( $\neg$ Adult  $\sqcup$  Human)(Andy),  
 $\neg$ Baby(Andy)

Baby(Andy), Adult(Andy),  
Human(Andy),  
 $\neg$ Adult(Andy)

Baby(Andy), Adult(Andy),  
Human(Andy),  
 $\neg$ Adult(Andy)

Baby(Andy), Adult(Andy),  
Human(Andy),  
 $\neg$ Baby(Andy)

Nr	Aussage	Aktion
5	(C $\sqcup$ D)(a)	Teile das Tableau in T1 und T2. Füge C(a) zu T1, D(a) zu T2 hinzu

# Aufgabe 1

- In diesem Fall werden alle Zweige des Tableaus zum Widerspruch geführt
- Damit existiert insgesamt ein Widerspruch!

# Aufgabe 2

- Das Datenset Linked Open Numbers enthält die Definition für 1 Milliarde Zahlen, inklusive ihrer Vorgänger und Nachfolger und weiterer Informationen, wie den Primfaktoren.
  - Definieren Sie eine Klasse für gerade Zahlen in OWL.
  - Definieren Sie eine Klasse für ungerade Zahlen in OWL.
  - Definieren Sie eine Klasse für Primzahlen in OWL.
  - Können Sie auch eine Klasse für Nicht-Primzahlen definieren?
  - Finden Sie einen effizienten(!) Weg, die paarweise Verschiedenheit dieser Zahlen zu definieren.

# Aufgabe 2

- Definieren Sie eine Klasse für gerade Zahlen in OWL

```
EvenNumber owl:equivalentClass [  
  a owl:Restriction;  
  owl:onProperty primefactor;  
  owl:someValuesFrom [owl:oneOf (:2)] ] .
```

# Aufgabe 2

- Definieren Sie eine Klasse für ungerade Zahlen in OWL

```
:UnevenNumber owl:equivalentClass [  
  a owl:Restriction;  
  owl:onProperty next;  
  owl:someValuesFrom :EvenNumber ] .
```

# Aufgabe 2

- Definieren Sie eine Klasse für Primzahlen in OWL

```
:PrimeNumber owl:equivalentClass [  
  a owl:Restriction;  
  owl:onProperty primefactor;  
  owl:hasSelf "true"^^xsd:boolean ] .
```

# Aufgabe 2

- Können Sie auch eine Klasse für Nicht-Primzahlen definieren?
- Das ist auf Basis der gegebenen Axiome nicht möglich
  - Wegen Open World Assumption
  - Nicht-gegebene Primfaktoren sind nicht automatisch falsch
  - Man müsste also die Zahlendefinitionen selbst ändern

# Aufgabe 2

- Definieren Sie paarweise Verschiedenheit

- Was haben wir schon?

```
:next a owl:ObjectProperty ;  
      rdfs:domain :Number ;  
      rdfs:range :Number .
```

- Was können wir damit machen?

```
:smallerThan a owl:ObjectProperty ,  
              owl:TransitiveProperty .  
:next rdfs:subPropertyOf :largerThan .
```

# Aufgabe 2

- Damit folgt schon mal die allgemeine "kleiner-als"-Relation aus den Nachfolgern:

:2 :next :3 .

:3 :next :4 .

:2 :lessThan :3 .

:3 :lessThan :4 .

:2 :lessThan :4 .

# Aufgabe 2

- Definieren wir jetzt noch folgendes:

```
:lessThan rdfs:subPropertyOf owl:differentFrom .
```

- Damit folgt zusätzlich

```
:2 owl:differentFrom :4 .
```

# Aufgabe 2

- Jetzt müssen wir uns noch um den anderen Fall kümmern (erste Zahl größer als die zweite)

```
:largerThan a owl:ObjectProperty .  
            owl:inversePropertyOf :previous .  
            rdfs:subPropertyOf owl:differentFrom .
```

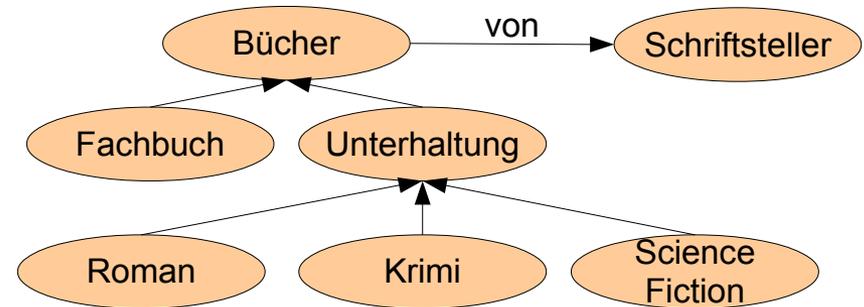
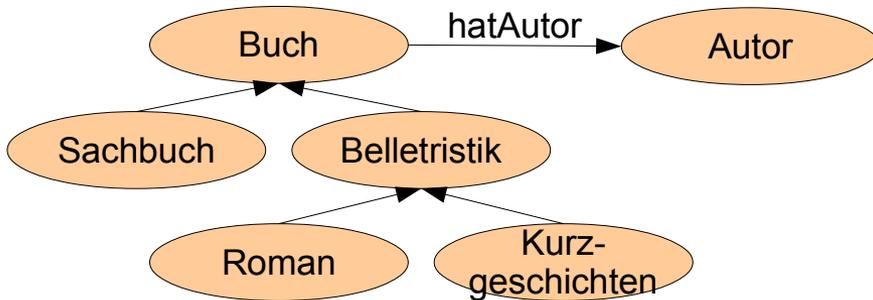
```
:previous a owl:ObjectProperty ;  
          owl:inversePropertyOf :next .  
          rdfs:subPropertyOf :largerThan .
```

# Aufgabe 2

- Der Test: alles live in Protégé!

# Aufgabe 3

- Gegeben folgende Ontologien:



- Wie können folgende Abbildungen gefunden werden:
  - Buch = Bücher
  - Autor = Schriftsteller
  - Sachbuch = Fachbuch
  - Belletristik = Unterhaltung
  - Roman = Roman
  - von = hatAutor

# Aufgabe 3

- Buch = Bücher
  - Edit-Distanz: 0.5 (nicht besonders hoch, vgl. 0.6 für Buch/Fisch)
  - 3-Gramme: 0
- Mögliche Lösung: Stemming
  - Buch → Buch
  - Bücher → Buch
- Autor = Schriftsteller
  - mit String-Methoden nicht findbar
  - mögliche Lösung: Synonymlexikon

# Aufgabe 3

- Sachbuch = Fachbuch
  - mit String-Methode gut findbar:
    - Edit-Distanz: 0.125
    - 3-Gramme: 83%
  - aber ist das nicht eher ein Glückstreffer?
  - vgl. *Handbuch* und *Handtuch*
    - Edit-Distanz: auch 0.125

# Aufgabe 3

- Belletristik = Unterhaltung
  - z.B. Synonymlexikon
  - Bounded Path Matching mit
    - Buch = Bücher und
    - Roman = Roman
- Roman = Roman
  - trivial
- von = hatAutor
  - über strukturbasiertes Verfahren aus
    - Buch = Bücher und
    - Autor = Schriftsteller

# Aufgabe 4

- Gegeben drei Häuser, ein pinkes, ein gelbes, und ein blaues, die nebeneinander stehen und von 1-3 nummeriert sind. In jedem der Häuser wohnt eine Person. Jede der Personen hat ein anderes Hobby (Rugby, Lacrosse oder Tennis) und ein anderes Lieblingsessen (Eier, Waffeln oder Eis). Gegeben sind darüber hinaus noch folgende Fakten:
  - Die Person, die gern Eis isst, lebt im pinken Haus.
  - Der Tennisspieler wohnt links vom Lacrossespieler.
  - Die Person in Haus drei spielt nicht Lacrosse.
  - Die Person, die Tennis spielt, wohnt im gelben Haus.
  - Das blaue Haus steht rechts vom gelben Haus.
  - Der Lacrossespieler wohnt direkt neben der Person, die Eier isst.

# Aufgabe 4

- Recap: wir hatten schon definiert
  - disjunkte Klassen für Häuser (linkes, mittleres rechtes, sowie pinkes, gelbes, blaues), Hobbies und Essen
  - Zuordnung zwischen Personen und Häusern, Hobbies, Essen
    - Funktional und invers funktional
    - Jede Person hat genau ein Haus, Hobby, Essen
  - Zuordnung von Nachbarhäusern

# Aufgabe 4

- Was wir zusätzlich brauchen
  - Häuser stehen links und rechts voneinander

```
:leftOf rdfs:subPropertyOf :neighborOf .
:rightOf rdfs:subPropertyOf :neighborOf .
:leftOf owl:inversePropertyOf :rightOf .
```
  - Jedes Haus kann nur *ein* linkes/rechtes Nachbarhaus haben

```
:leftOf a owl:FunctionalProperty,
          owl:InverseFunctionalProperty .
:rightOf a owl:FunctionalProperty,
          owl:InverseFunctionalProperty .
```
  - Instanzbeziehungen:

```
:linkesHaus :leftOf :mittleresHaus .
:mittleresHaus :leftOf :rechtesHaus.
```

# Aufgabe 4

- Formalisierung komplexerer Axiome
  - *Der Tennisspieler wohnt links vom Lacrossespieler.*
  - Hierfür brauchen wir zwei Personen und zwei Häuser:

```
:p1a :livesIn :h1a ;  
      :plays :Tennis ;  
      owl:differentFrom :p1b .  
:p1b :livesIn :h2a ;  
      :plays :lacrosse .  
:h1a :leftOf :h2a ;  
      owl:differentFrom :h2a .
```

# Aufgabe 4

- Das reicht offenbar noch nicht
  - Der Reasoner findet noch nicht alle Lösungen
  - Wahrscheinlich haben wir noch irgend etwas vergessen
    - d.h.: wir nehmen etwas implizit an, was wir nicht explizit gemacht haben
    - Ideen?

# Aufgabe 4

- Angenommen
  - Jemand wohnt im linken Haus
  - hat einen Nachbarn, der Lacrosse spielt
  - und einen Nachbarn, der im blauen Haus wohnt
- → der Lacrosse-Spieler wohnt im blauen Haus
- Warum?
  - weil das linke und das rechte Haus nur je ein Nachbarhaus haben

# Aufgabe 4

- Das brauchen wir noch!

```
House owl:disjointUnionOf (:HouseWithOneNeighbor
                             :HouseWithTwoNeighbors)

:HouseWithOneNeighbor rdfs:subClassOf
  :House ,
  [ a owl:Restriction ;
    owl:onProperty :hasNeighbor ;
    owl:cardinality 1^^xsd:integer ] .

:HouseWithTwoNeighbors rdfs:subClassOf
  :House ,
  [ a owl:Restriction ;
    owl:onProperty :hasNeighbor ;
    owl:cardinality 2^^xsd:integer ] .
```

# Aufgabe 4

- Ob Du wirklich richtig stehst,  
siehst Du, wenn der Reasoner angeht...

# Fertig für dieses Jahr!



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT



# Vorlesung Semantic Web



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

Vorlesung im Wintersemester 2012/2013

Dr. Heiko Paulheim

Fachgebiet Knowledge Engineering