

Vorlesung Semantic Web



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Vorlesung im Wintersemester 2011/2012

Dr. Heiko Paulheim

Fachgebiet Knowledge Engineering

Was bisher geschah...

- Betrachten wir folgenden Satz:
- "Madrid ist die Hauptstadt von Spanien."
- Aussagen, die wir erhalten können:
 - "Madrid ist die Hauptstadt von Spanien." ✓
 - "Spanien ist ein Land." ✓
 - "Madrid ist eine Stadt." ✓
 - "Madrid liegt in Spanien." ✓
 - "Barcelona ist nicht die Hauptstadt von Spanien." ✗
 - "Madrid ist nicht die Hauptstadt von Frankreich." ✗
 - ...

Was bisher geschah...

- Wir haben gelernt, was Ontologien sind
 - RDF Schema als Sprache für Ontologien
- Mit RDF Schema können wir schon interessante Dinge tun
 - aber manche Dinge eben nicht
 - Kardinalitäten festlegen
 - Widersprüche erzeugen
 - Die Non Unique Naming Assumption "austricksen"
 - Die Open World Assumption "austricksen"
 - ...

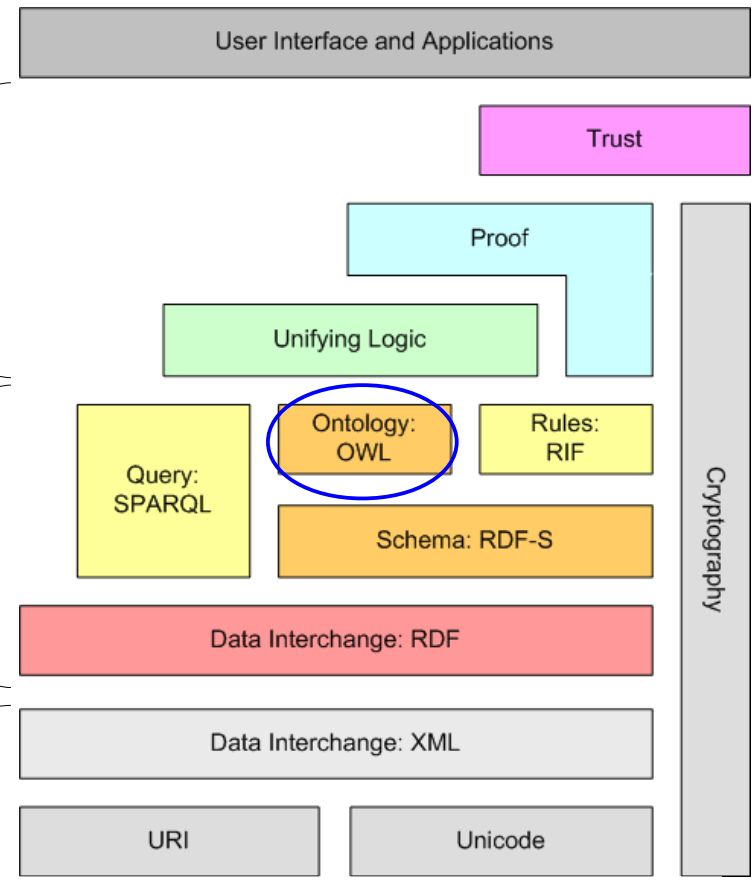
Semantic Web – Aufbau



here be dragons...

Semantic-Web-
Technologie
(Fokus der Vorlesung)

Technische
Grundlagen



Berners-Lee (2009): *Semantic Web and Linked Data*
<http://www.w3.org/2009/Talks/0120-campus-party-tbl/>

Web Ontology Language (OWL)



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- Warum eigentlich *OWL* und nicht *WOL*?



Web Ontology Language (OWL)

- Ausdrucksmächtiger als RDF Schema
- W3C-Standard (2004)
- Trade-off:
 - Ausdrucksmächtigkeit
 - Komplexität für Reasoner
 - Entscheidbarkeit
- Lösung: drei Varianten für OWL
 - OWL Lite
 - OWL DL
 - OWL Full

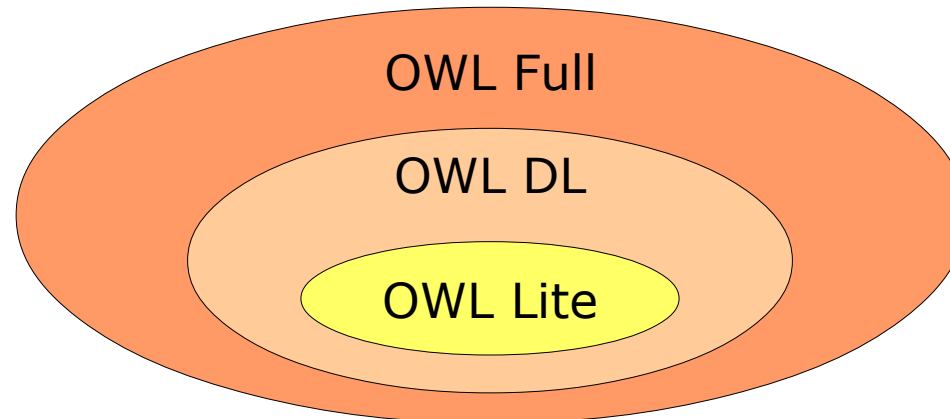


Web Ontology Language (OWL)



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- Drei Varianten
 - steigende Ausdrucksmächtigkeit
 - abwärtskompatibel



OWL und RDF Schema

- beide basieren auf RDF
 - Auch OWL-Ontologien lassen sich in RDF ausdrücken!
 - als Tripel oder XML
- OWL ist ausdrucksmächtiger als RDF Schema
- Kompatibilität
 - OWL Lite und OWL DL nicht voll kompatibel zu RDF Schema
 - verwenden aber Teile von RDF Schema wieder
 - OWL Full und RDF Schema sind kompatibel

- Grundkonzept ist eine Klasse (`owl:Class`)
- Subklassenbeziehung wie bisher mit `rdfs:subClassOf`
 - Insbesondere gilt:
`owl:Class rdfs:subClassOf rdfs:Class .`
- Zwei vordefinierte Klassen:
 - `owl:Thing`
 - `owl:Nothing`
- Für jede Klasse `c` gilt:
 - `c rdfs:subClassOf owl:Thing .`
 - `owl:Nothing rdfs:subClassOf c .`

- Klassen können auch als Schnittmengen bekannter Klassen entstehen

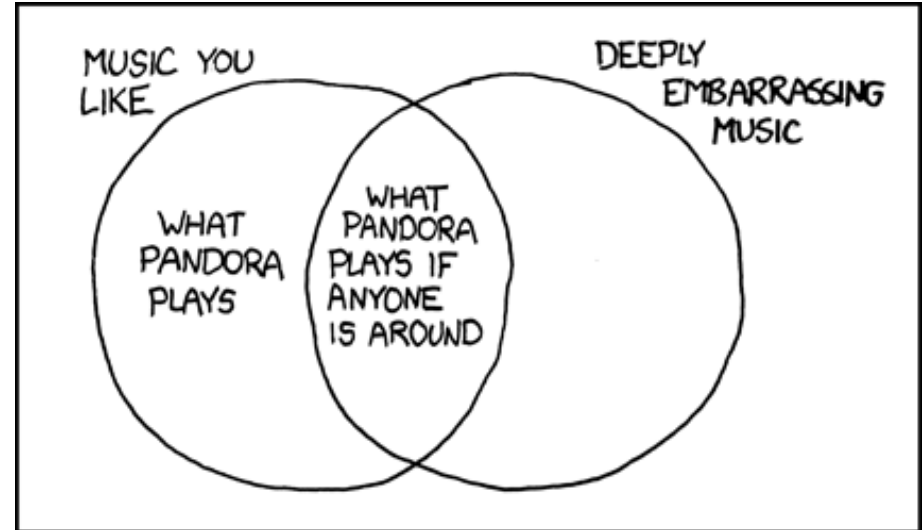
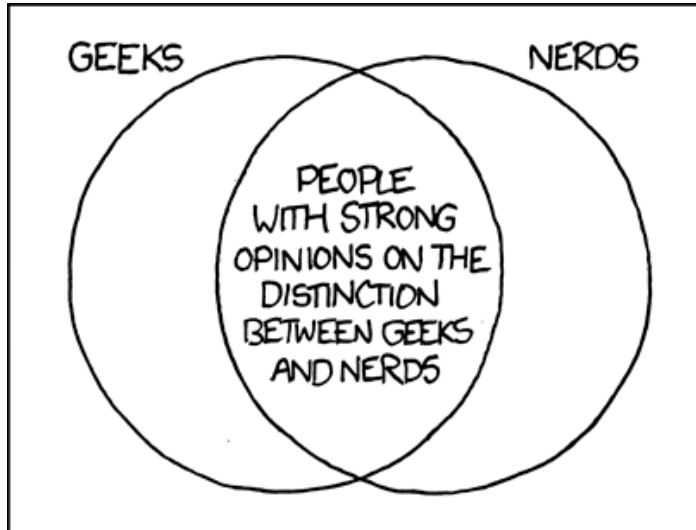
```
:SwimmingMammals owl:intersectionOf  
  (:SwimmingAnimals :Mammals) .
```

- Vereinigungs- und andere Mengen gibt es auch
 - aber nicht in OWL Lite!

OWL:Klassen



TECHNISCHE
UNIVERSITÄT
DARMSTADT



<http://xkcd.com/747/>
<http://xkcd.com/668/>

OWL: Properties

- Auch in RDF/S gibt es `rdf:Property`
 - aber keine Unterscheidung in Datenwerte und Relationen:

```
:name a rdf:Property .  
:name rdfs:range xsd:string .
```

```
:knows a rdf:Property .  
:knows rdfs:range foaf:Person .
```

- In OWL unterscheidet man
 - `owl:DatatypeProperty`
 - `owl:ObjectProperty`

- Es gilt:
 - `owl:DatatypeProperty rdfs:subClassOf rdf:Property .`
 - `owl:ObjectProperty rdfs:subClassOf rdf:Property .`

- Für `DatatypeProperty` werden wieder XML-Datentypen genutzt
 - prinzipiell können alle verwendet werden
 - Tools müssen mindestens `xsd:string` und `xsd:integer` unterstützen

- Properties können auch in OWL Hierarchien bilden

```
:capitalOf rdfs:subPropertyOf :locatedIn .
```

- Domain und Range können angegeben werden

- Domain

- nur Klassen

```
:name rdfs:domain foaf:Person .
```

- Range

- von DataProperties: XML-Datentypen

```
:name rdfs:range xsd:string .
```

- von ObjectProperties: Klassen oder Restriktionen*

```
:knows rdfs:range foaf:Person .
```

* lernen wir noch kennen

Gleichheit und Ungleichheit

- Gleichheit zwischen Individuen
 - Erlaubt den Umgang mit mehreren Definitionen
 - auch in unterschiedlichen Datensets
 - behebt Probleme mit Non unique naming assumption

```
:Muenchen owl:sameAs :Munich .
```

- Gleichheit zwischen Klassen und Properties
 - Ermöglicht Beziehungen zwischen Datensets auf T-Box-Ebene
 - erlaubt komplexere Konstrukte (sehen wir später)

```
:UniversityTeachers owl:equivalentClass :Lecturers .  
:teaches owl:equivalentProperty :lecturerFor .
```

Gleichheit und Ungleichheit

- Ungleichheit zwischen Individuen
 - Ermöglicht fortgeschrittenes Reasoning
 - wie wir bald sehen werden

```
:Muenchen owl:differentFrom :Hamburg .
```

- Einfachere Notation für viele Individuen:
 - mit Hilfe von Listen
 - siehe *Notation von Listen in RDF*

```
owl:AllDifferent owl:distinctMembers  
  (:Munich :Hamburg :Berlin :Darmstadt :Kassel) .
```


OWL: Besondere Properties

▪ Symmetrische Properties

```
:sitsOppositeOf a owl:SymmetricProperty .  
    :Tom :sitsOppositeOf :Sarah .  
→    :Sarah :sitsOppositeOf :Tom .
```

▪ Inverse Properties

```
:supervises owl:inverseOf :supervisedBy .  
    :Tom :supervises :Julia .  
→    :Julia :supervisedBy :Tom .
```

▪ Transitive Properties

```
:hasOfficeMate a owl:TransitiveProperty .  
    :Tom :hasOfficeMate :Jon . :Jon :hasOfficeMate :Kim .  
→    :Tom :sharesOfficeWith :Kim .
```

Einschränkungen für besondere Properties

- Nur ObjectProperties dürfen transitiv, symmetrisch und invers sein
 - DataProperties nicht
- Warum?
- *Previously on RDF:*
 - "Literele können nur Objekt sein, nicht Subjekt oder Prädikat"

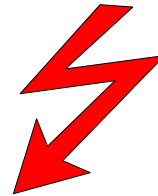
Einschränkungen für besondere Properties

- Angenommen,

```
:samePerson a owl:DatatypeProperty .  
:samePerson rdfs:range xsd:string .  
:samePerson a owl:SymmetricProperty .
```

```
:Peter :samePerson "Peter " .
```

→ "Peter" :samePerson :Peter .



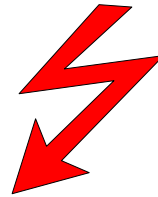
Einschränkungen für besondere Properties

- Angenommen,

```
:hasName a owl:DatatypeProperty .  
:hasName rdfs:range xsd:string .  
:hasName owl:inverseOf :nameOf .
```

```
:Peter :hasName "Peter " .
```

→ "Peter" :nameOf :Peter .



Einschränkungen für besondere Properties

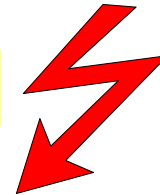
- Auch `owl:TransitiveProperty` ist auf `ObjectProperties` beschränkt

```
:hasPseudonym a owl:DatatypeProperty .  
:hasPseudonym rdfs:range xsd:string .  
:hasPseudonym a owl:TransitiveProperty .
```

```
:Thomas :hasPseudonym "Dr. Evil" .
```

```
+ "Dr. Evil" :hasPseudonym "Skullhead" .
```

```
→ :Thomas :hasPseudonym "Skullhead" .
```



- Was müsste hier stehen, damit wir die Aussage mit `owl:TransitiveProperty` folgern könnten?

Funktionale Properties

▪ Funktionale Properties

```
:hasCapital a owl:FunctionalProperty .  
:Finland :hasCapital :Helsinki .  
:Finland :hasCapital :Helsingfors .  
→ :Helsinki owl:sameAs :Helsingfors .
```

▪ Funktionale Properties

- wenn A zu B in Beziehung fp steht
- und A zu C in Beziehung fp steht
- dann sind B und C gleich
- vereinfacht: $fp(x)$ hat nur einen eindeutigen Wert
- "es kann nur einen geben"



<http://www.allmystery.de/dateien/uh60808,1274716100,highlander-christopher-lambert.jpg>

▪ Invers funktionale Properties

```
:capitalOf a owl:InverseFunctionalProperty .  
:Helsinki :capitalOf :Finland .  
:Helsingfors :capitalOf :Finland .  
→ :Helsinki owl:sameAs :Helsingfors .
```

▪ Invers funktionale Properties

- wenn A zu C in Beziehung ifp steht
- und B zu C in Beziehung ifp steht
- dann sind A und B gleich
- Vereinfacht: ifp(x) identifiziert x eindeutig
 - wie ein Primärschlüssel in einer Datenbank

Invers funktionale Properties

- Invers funktionale Properties
 - äquivalent zu: die Inverse ist ein funktionales Property
- Am Beispiel:
 - `:capitalOf a owl:InverseFunctionalProperty .`
- ist äquivalent zu
 - `:capitalOf owl:inverseOf :hasCapital .`
 - `:hasCapital a owl:FunctionalProperty .`

Invers funktionale Properties



- Warum gilt das? Einerseits haben wir

```
:capitalOf a owl:InverseFunctionalProperty .
```

```
:Helsinki :capitalOf :Finland .
```

```
:Helsingfors :capitalOf :Finland .
```

```
→ :Helsinki owl:sameAs :Helsingfors .
```

- und andererseits

```
:capitalOf owl:inverseOf :hasCapital .
```

```
:hasCapital a owl:FunctionalProperty .
```

```
:Finland :hasCapital :Helsinki .
```

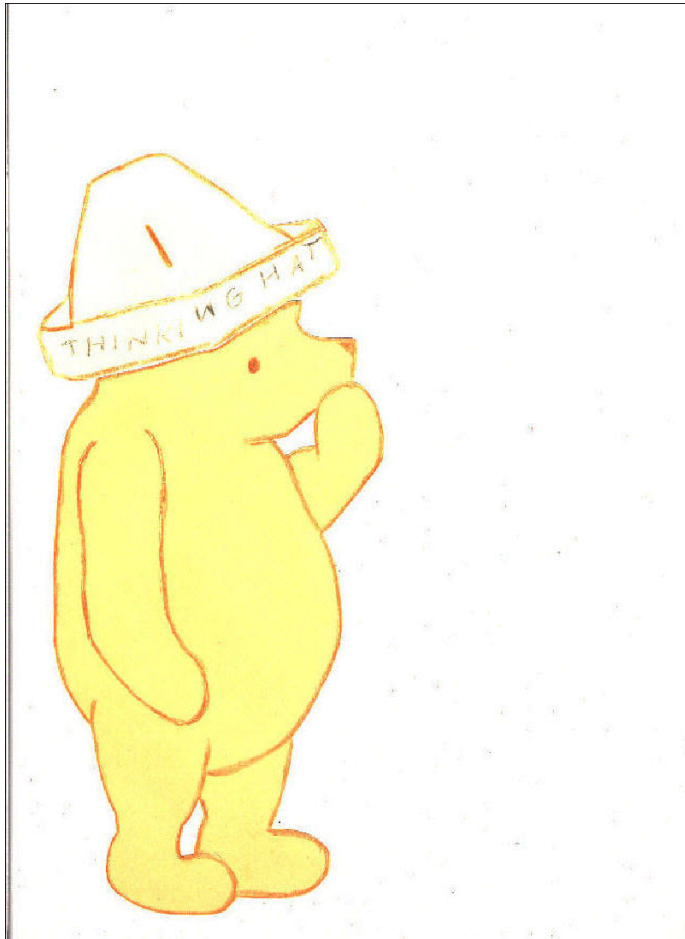
```
:Finland :hasCapital :Helsingfors .
```

```
→ :Helsinki owl:sameAs :Helsingfors .
```

Puh!



TECHNISCHE
UNIVERSITÄT
DARMSTADT



- OWL kann noch viel mehr...
- ...aber mit dem, was wir bis jetzt haben, können wir auch schon einiges erreichen
- Zurück zu unserem Beispiel...

Was bisher geschah...

- Betrachten wir folgenden Satz:
- "Madrid ist die Hauptstadt von Spanien."
- Aussagen, die wir erhalten können:
 - "Madrid ist die Hauptstadt von Spanien." ✓
 - "Spanien ist ein Land." ✓
 - "Madrid ist eine Stadt." ✓
 - "Madrid liegt in Spanien." ✓
 - "Barcelona ist nicht die Hauptstadt von Spanien." ✗
 - "Madrid ist nicht die Hauptstadt von Frankreich." ✗
 - ...

Ausdrucksstarke Ontologien mit OWL



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- "Barcelona ist nicht die Hauptstadt von Spanien." ✖
- Warum eigentlich nicht?
 - Länder haben nur genau eine Hauptstadt
 - Barcelona ist nicht gleich Madrid
- Also:

```
:capitalOf a owl:InverseFunctionalProperty .  
:Madrid :capitalOf :Spain .  
:Madrid owl:differentFrom :Barcelona .
```

```
ASK { :Barcelona :capitalOf :Spain . } → false
```

Ausdrucksstarke Ontologien mit OWL



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- "Madrid ist nicht die Hauptstadt von Frankreich." ✖
- Warum eigentlich nicht?
 - Eine Stadt kann nur Hauptstadt von einem Land sein
 - Spanien ist nicht gleich Frankreich
- Also:

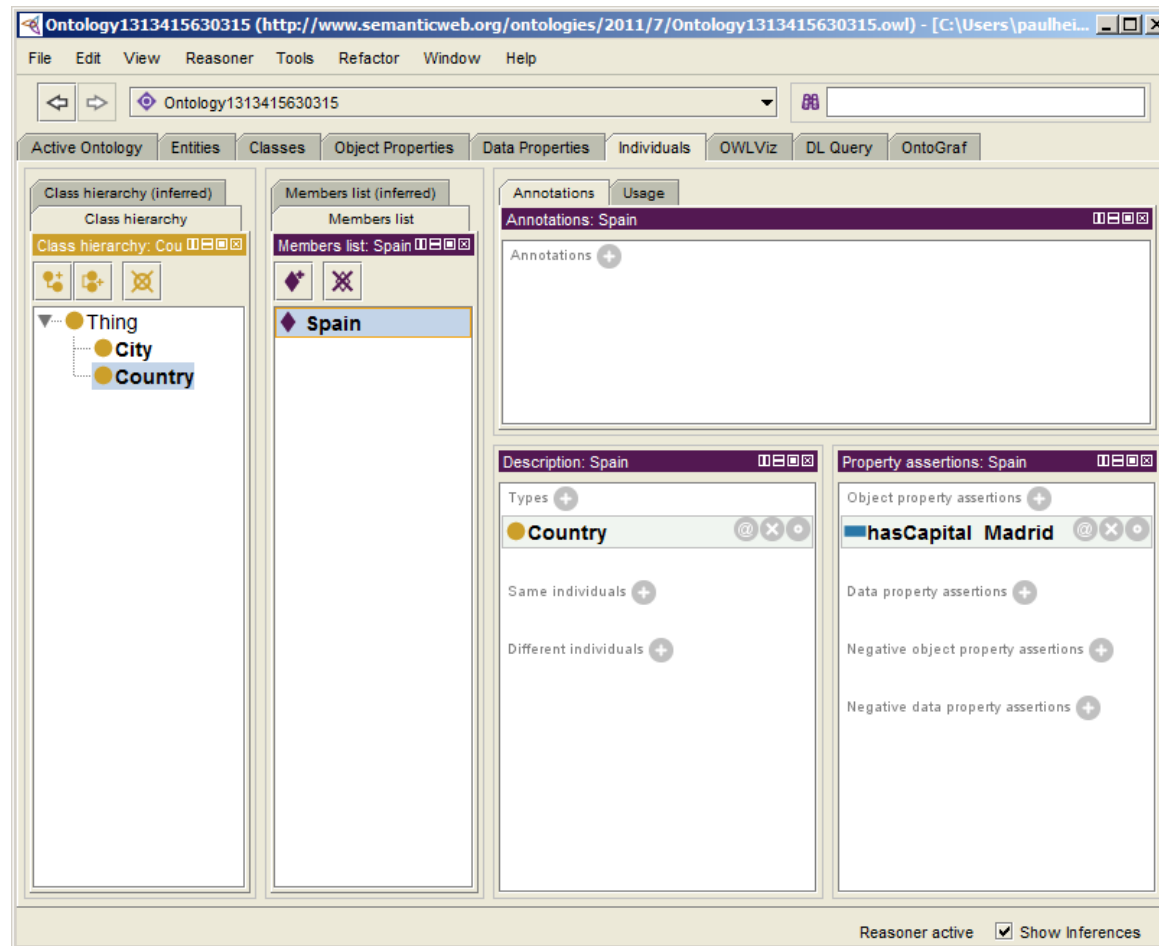
```
:capitalOf a owl:FunctionalProperty .  
:Madrid :capitalOf :Spain .  
:Spain owl:differentFrom :France .
```

```
ASK { :Madrid :capitalOf :France . } → false
```

Protégé – ein Editor für OWL



TECHNISCHE
UNIVERSITÄT
DARMSTADT



OWL – The Story so Far

- Definition von Mengen von Dingen
 - Vereinigung, Schnittmenge, ...
- Properties
 - symmetrisch, invers, transitiv, funktional, invers funktional

- Restriktionen
 - sind ein sehr wichtiges und mächtiges Sprachmittel
 - Beispiel: vegane Rezepte enthalten nur Gemüse als Zutaten

```
:VeganRecipe rdfs:subClassOf :Recipe .  
:VeganRecipe rdfs:subClassOf [  
  a owl:Restriction .  
  owl:onProperty :hasIngredient .  
  owl:allValuesFrom :Vegetable .  
] .
```


Unterschied von Restriktionen und Wertebereichen



- Restriktion: lokal für eine Klasse

```
:VeganRecipe rdfs:subClassOf [  
  a owl:Restriction ;  
  owl:onProperty :hasIngredient ;  
  owl:allValuesFrom :Vegetable .  
] .
```

- es kann andere Verwendungen von `hasIngredient` geben, die auch Fleisch oder Fisch vorsehen

- Wertebereich: globale Einschränkung

```
:hasIngredient rdfs:range :Food .
```

- das gilt für *alle* Verwendungen von `hasIngredient`.

Aufbau von Restriktionen

- onProperty
 - gibt an, auf welches Property sich die Restriktion bezieht
- Restriktion der Werte
 - owl:allValuesFrom – alle Werte müssen dieser Klassen angehören
 - owl:someValuesFrom – mindestens ein Wert muss dieser Klasse angehören
- Restriktion der Kardinalität
 - owl:minCardinality – mindestens n Werte
 - owl:maxCardinality – maximal n Werte
 - owl:cardinality – genau n Werte
- Kardinalitäts- und Wertrestriktion nicht kombinierbar!

OWL Lite: nur $n=0$
oder $n=1$

Weitere Beispiele für Restriktionen



- Jeder Mensch hat genau eine Mutter

```
:Human rdfs:subClassOf [  
  a owl:Restriction ;  
  owl:onProperty :hasMother ;  
  owl:cardinality 1^^xsd:integer .  
] .
```

- Fahrräder sind Fahrzeuge ohne Motor

```
:Bicycle rdfs:subClassOf :Vehicle .  
:Human rdfs:subClassOf [  
  a owl:Restriction ;  
  owl:onProperty :hasMotor ;  
  owl:cardinality 0^^xsd:integer .  
] .
```

Weitere Beispiele für Restriktionen



- Für Ballsportarten braucht man einen Ball

```
:BallSports rdfs:subClassOf [  
  a owl:Restriction ;  
  owl:onProperty :requires ;  
  owl:someValuesFrom :Ball .  
]
```

- Alle Sportarten, für die man einen Ball braucht, sind Ballsportarten

```
:BallSports owl:equivalentClass [  
  a owl:Restriction ;  
  owl:onProperty :requires ;  
  owl:someValuesFrom :Ball .  
]
```

- Was ist da jetzt eigentlich der Unterschied?

Klassendefinitionen mit Restriktionen



- Gegeben:

```
:BallSports owl:equivalentClass [  
    a owl:Restriction ;  
    owl:onProperty :requires ;  
    owl:someValuesFrom :Ball .  
] .  
  
:Soccer :requires :soccerBall .  
:soccerBall a :Ball.
```

- Daraus kann der Reasoner schließen, dass Fußball ein Ballspiel ist.
- Das geht mit `subClassOf` nicht
- Allerdings: Gymnastikübungen mit Ball werden so auch als Ballsport erkannt...

Restriktionen als Wertebereiche

- Restriktionen können nicht nur Klassen definieren
- Beispiel: man kann Bücher, Zeitschriften, Plakate lesen
 - also alles, was Buchstaben enthält

- Wertebereich des Prädikats *lesen*:

```
:reads rdfs:range [  
  a owl:Restriction ;  
  owl:onProperty :containsLetter ;  
  owl:minCardinality 1^^xsd:integer .  
] .
```

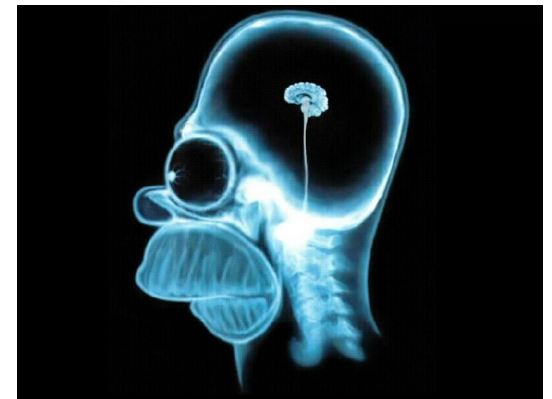


Restriktionen als Definitionsbereiche

- Was für Wertebereiche geht, geht auch für Definitionsbereiche
- z.B.: um über etwas nachzudenken, braucht man ein Gehirn

- Domäne von *thinksAbout*:

```
:thinksAbout rdfs:domain [  
    a owl:Restriction ;  
    owl:onProperty :hasBodyPart ;  
    owl:someValuesFrom :Brain .  
]
```



Restriktionen schachteln



- Es geht immer auch etwas komplexer
- z.B. Großeltern sind Eltern von mindestens einem Menschen, der mindestens ein Kind hat:

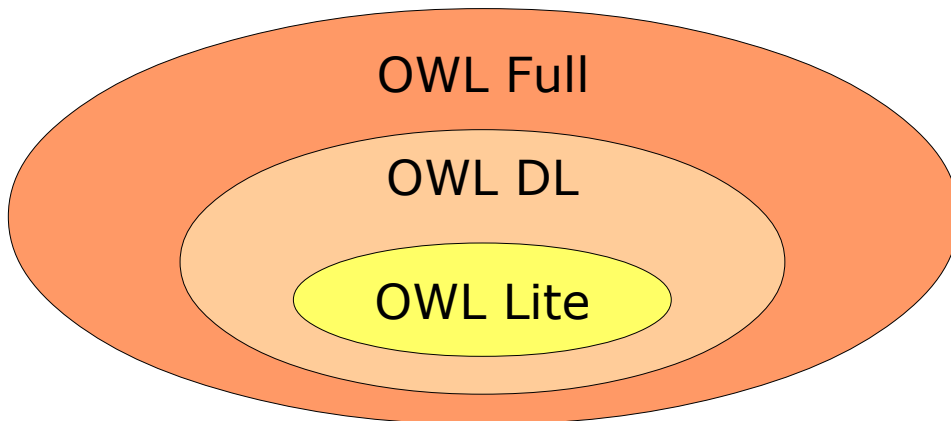
```
:GrandParent owl:equivalentClass [  
  a owl:Restriction ;  
  owl:onProperty :hasChild ;  
  owl:someValuesFrom [  
    a owl:Restriction ;  
    owl:onProperty :hasChild ;  
    owl:minCardinality 1^^xsd:integer .  
  ] .  
]
```


Restriktionen

- in Protégé anlegen:
 - über Texteingabe im ClassExpression Editor
 - Syntax-Details siehe Tutorial
- ...jetzt live on Protégé!

Web Ontology Language (OWL)

- Was wir bisher kennen gelernt haben
 - ist das Vokabular von OWL Lite
 - schon einigermaßen brauchbar
 - "A little semantics goes a long way."



Darf's etwas mehr sein?



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- OWL Lite kann schon viel
- OWL DL und OWL Full können mehr
 - sind aber auch komplexer
 - machen Reasoning schwerer

- DL steht für "Description Logics"
 - Beschreibungslogik
 - eine Untermenge der Prädikatenlogik erster Stufe
- damit werden wir uns beim Reasoning noch beschäftigen

Komplexere Mengendefinitionen



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- Vereinigungsmenge

`:FacultyMembers owl:unionOf (:Students, :Professors) .`

- Komplementäre Menge

`:LivingThings owl:complementOf :InanimateThings .`

- Disjunkte Mengen

`:EdibleMushrooms owl:disjointWith :PoisonousMushrooms .`

Komplexere Mengendefinitionen



- Auch zusammen mit Restriktionen möglich

```
:VegetarianRecipe rdfs:subClassOf [  
  a owl:Restriction ;  
  owl:onProperty :hasIngredient ;  
  owl:allValuesFrom [  
    a owl:Class .  
    owl:complementOf :Meat.  
  ]  
] .
```

Coming Soon

- weitere Sprachmittel in OWL DL und Full
 - noch komplexere Ontologien
- warum OWL Lite/DL und RDF Schema eigentlich inkompatibel sind
- Neuerungen in OWL 2

Semantic Web – Aufbau



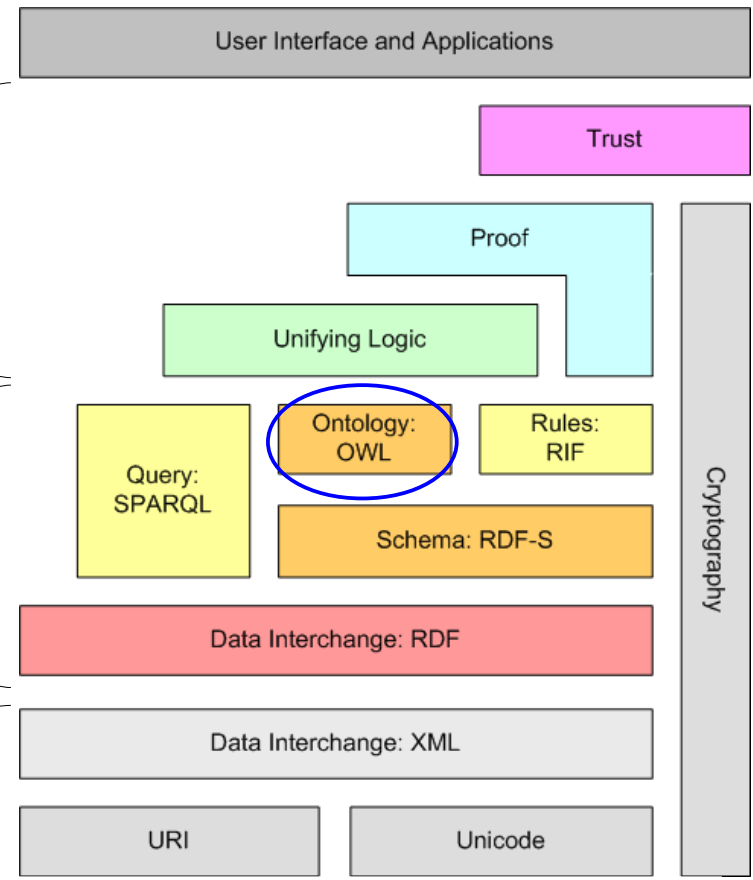
TECHNISCHE
UNIVERSITÄT
DARMSTADT



here be dragons...

Semantic-Web-
Technologie
(Fokus der Vorlesung)

Technische
Grundlagen



Berners-Lee (2009): *Semantic Web and Linked Data*
<http://www.w3.org/2009/Talks/0120-campus-party-tbl/>

Vorlesung Semantic Web



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Vorlesung im Wintersemester 2011/2012

Dr. Heiko Paulheim

Fachgebiet Knowledge Engineering