

Vorlesung Semantic Web



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Vorlesung im Wintersemester 2011/2012

Dr. Heiko Paulheim

Fachgebiet Knowledge Engineering

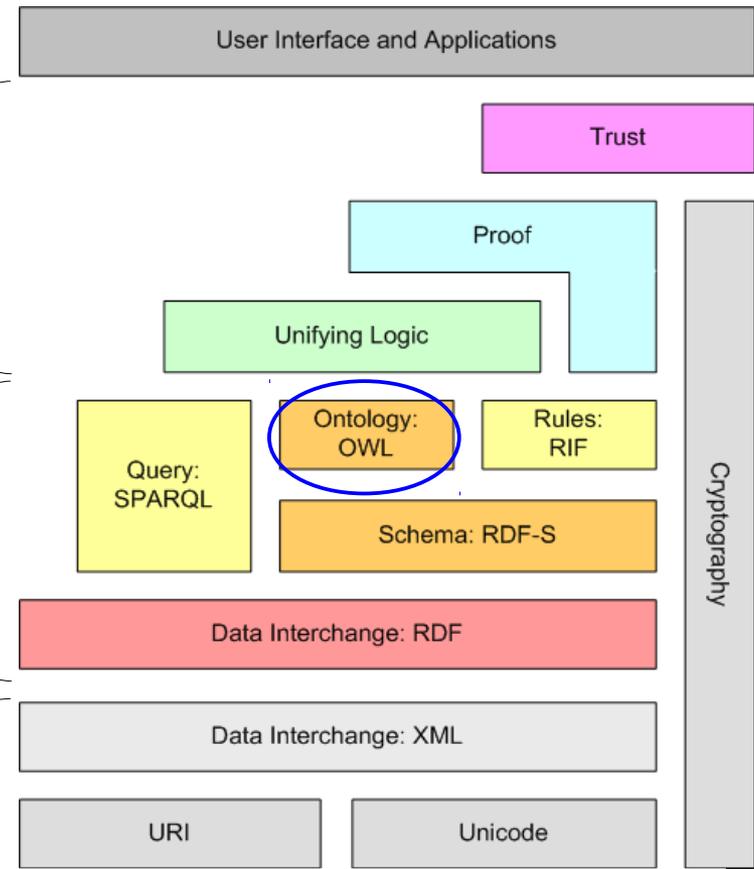
Semantic Web – Aufbau



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/>

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

- Sprachmittel von OWL
 - Properties: symmetrisch, transitiv, ...
 - Klassen: Schnittmengen, Komplement, Vereinigung
 - Restriktionen
- OWL kann aber noch mehr

Recap: Prinzipien von RDF

- Grundlegende Prinzipien der Wissensbereitstellung im Semantic Web
- AAA: Anybody can say Anything about Anything
- Non-unique name assumption
 - können wir umgehen mit owl:sameAs und owl:equivalentClass
- Open World Assumption
 - damit müssen wir bis jetzt leben

Abgeschlossene Klassen

- Bisher gilt die Open World Assumption
 - alles, was wir nicht wissen, *könnte* gelten
- Beispiel:
 - `:Heiko a :PeopleInD219 .`
 - `:Eneldo a :PeopleInD219 .`
 - `:Christian a :PeopleInD219 .`
 - Das heißt nicht, dass es nicht noch weitere Personen in D219 geben könnte, z.B.
 - `:KarlHeinz a :PeopleInD219 .`
- Manchmal will man aber genau das sagen...

Abgeschlossene Klassen

- Das geht mit `owl:oneOf`
- Beispiel:

```
:PeopleInD219 owl:oneOf (:Heiko :Eneldo :Christian) .
```
- Was bedeutet jetzt die Aussage

```
:KarlHeinz a :PeopleInD219 .
```


?

Einschränkungen von Wertebereichen

▪ Für Relationen:

```
:AfricanStates owl:subClassOf [  
  a owl:Restriction ;  
  owl:onProperty :locatedOnContinent  
  owl:hasValue :Africa ] .
```

▪ Für Datenwerte:

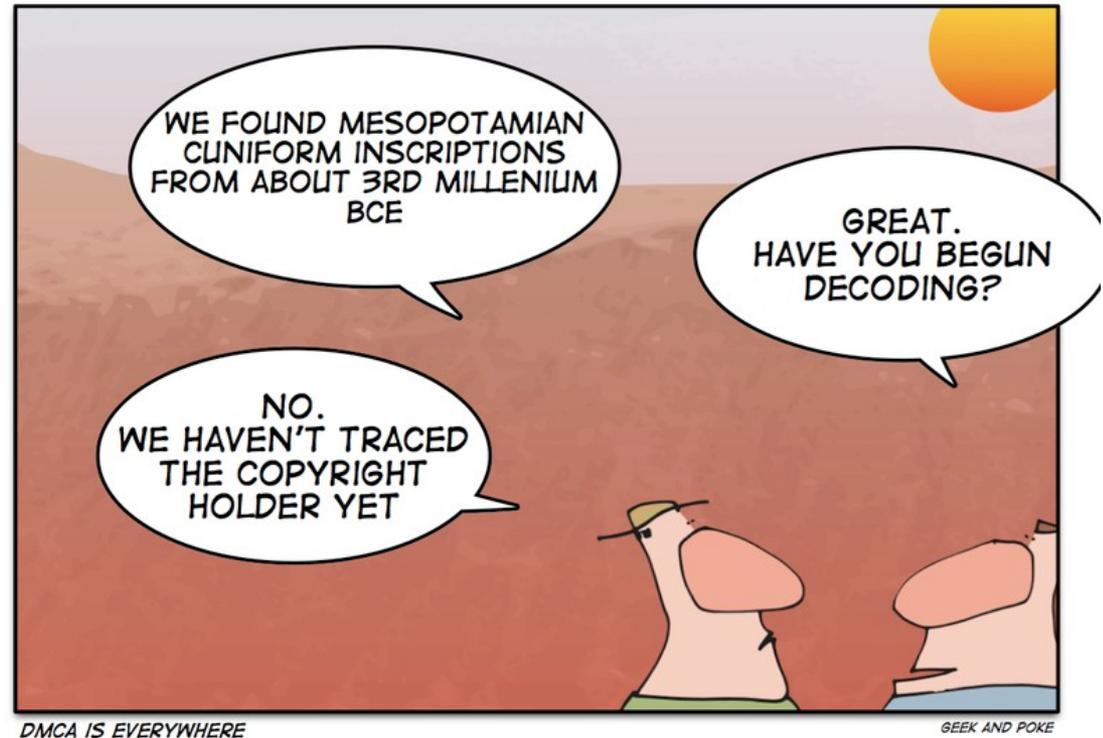
```
:AlbumsFromTheEarly80s owl:subClassOf [  
  a owl:Restriction ;  
  owl:onProperty :year  
  owl:dataRange  
    (1980^^xsd:integer  
     1981^^xsd:integer  
     1982^^xsd:integer) ] .
```

A Tale from the Road

- ALIS: EU-Projekt (2006-2009)
- Automated Legal Intelligent System
 - Automatische Suche nach relevanten Gesetzestexten
 - für einen bestimmten Fall
 - Mit Hilfe von Ontologien, Reasoning, etc.
 - Anwendungsfall: Copyright

A Tale from the Road

- Wichtige Unterscheidung (u.a.):
 - Single Author Work
 - Multi Author Work



http://geekandpoke.typepad.com/geekandpoke/2006/10/copyright_and_a.html

A Tale from the Road



▪ Naive Lösung:

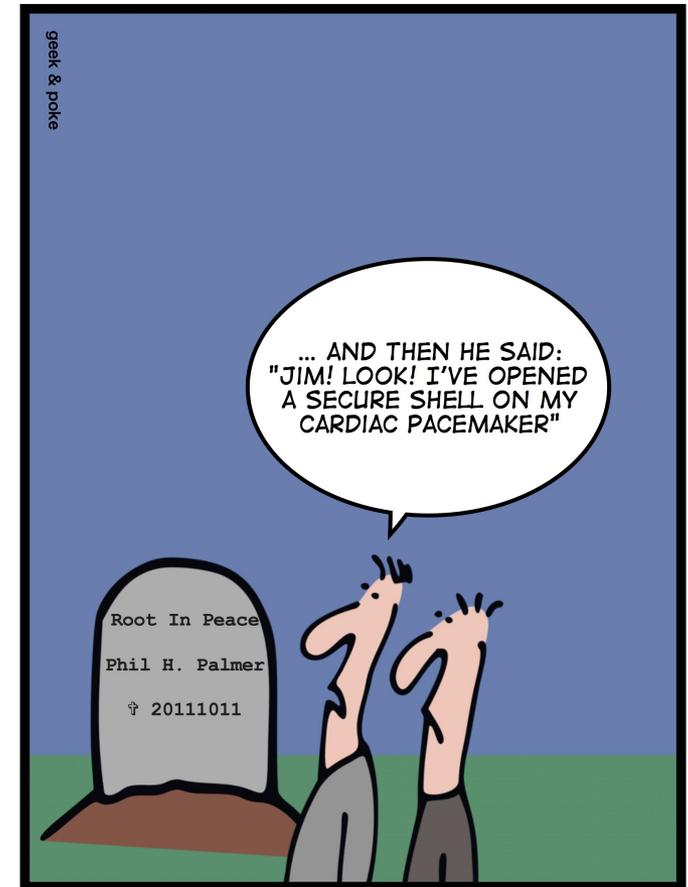
```
:hasAuthor a owl:ObjectProperty;  
           rdfs:domain :Work ;  
           rdfs:range  :Author .
```

```
:SingleAuthorWork rdfs:subClassOf  
  :Work,  
  [ a owl:Restriction;  
    owl:onProperty :hasAuthor ;  
    owl:cardinality 1^^xsd:integer ] .
```

```
:MultiAuthorWork rdfs:subClassOf  
  :Work,  
  [ a owl:Restriction;  
    owl:onProperty :hasAuthor ;  
    owl:minCardinality 2^^xsd:integer ] .
```

A Tale from the Road

- Ergebnis:
 - keine so gute Idee...
 - warum eigentlich nicht?



http://geekandpoke.typepad.com/geekandpoke/2006/10/copyright_and_a.html

- In OWL Lite und OWL DL dürfen Klassen nicht gleichzeitig (uneingeschränkt) als Instanzen verwendet werden
- In OWL Full gibt es diese Beschränkung nicht:
 - `:Elephant a owl:Class .`
 - `:Elephant a :Species .`
 - `:Elephant :livesIn :Africa .`
 - `:Species a owl:Class .`
- In OWL Lite/DL ist eine Klasse immer nur Instanz von `owl:Class`
- In OWL Lite/DL können Klassen nur eine definierte Menge von Relationen eingehen (z.B. `rdfs:subClassOf`).

Dinge, die man mit OWL Full machen kann

- Klassen dürfen ihrerseits als Subjekte und Objekte verwendet werden
- Zum Beispiel ginge folgendes:

```
:ElementaryClass rdfs:subClassOf owl:Class .
```

```
:ElementaryClass rdfs:subClassOf [  
  a owl:Restriction ;  
  owl:onProperty rdfs:subClassOf ;  
  owl:maxCardinality 0^^xsd:integer ] .
```

Weitere Sprachmöglichkeiten

- Metadaten zu Klassen und Properties definieren
- Geht mit `owl:AnnotationProperty`
 - `:creator a owl:AnnotationProperty .`
 - `:creationTime a owl:AnnotationProperty .`
 - `:Animal a owl:Class .`
 - `:Animal :creator :Heiko .`
 - `:Animal :creationTime "2011"^^xsd:integer .`
- **Achtung:** in OWL Lite/DL dürfen AnnotationProperties nicht mit anderen Properties vermischt werden!

Weitere Sprachmöglichkeiten

- Metadaten zur Ontologie

Zusammen mit einem xml-Namespace identifiziert das die aktuelle Ontologie

```
<owl:Ontology rdf:about="">  
  <owl:versionInfo>v 1.2 2011-08-23</owl:versionInfo>  
  <rdfs:comment>An example ontology</rdfs:comment>  
  <owl:imports rdf:resource="http://www.example.org/foo"/>  
</owl:Ontology>
```

- Weitere definierte Attribute:

```
owl:priorVersion  
owl:incompatibleWith  
owl:backwardCompatibleWith
```

Modulare Ontologien mit owl:import

- Die `owl:import` Anweisung wird von gängigen Werkzeugen aufgelöst
- Das ermöglicht die Wiederverwendung von Ontologien
- Beispiel: Entwicklung einer Studenten-Ontologie

```
@prefix : <http://students.org/> .
```

```
@prefix foaf: <http://xmlns.com/foaf/0.1/>
```

```
<http://students.org> a owl:Ontology .
```

```
<http://students.org> owl:imports
```

```
<http://xmlns.com/foaf/0.1/> .
```

```
:Student a foaf:Person .
```

RDF Schema vs. OWL

- OWL Lite und DL sind zu RDF Schema inkompatibel!
- Die Verwendung von Klassen ist in RDF Schema nicht eingeschränkt

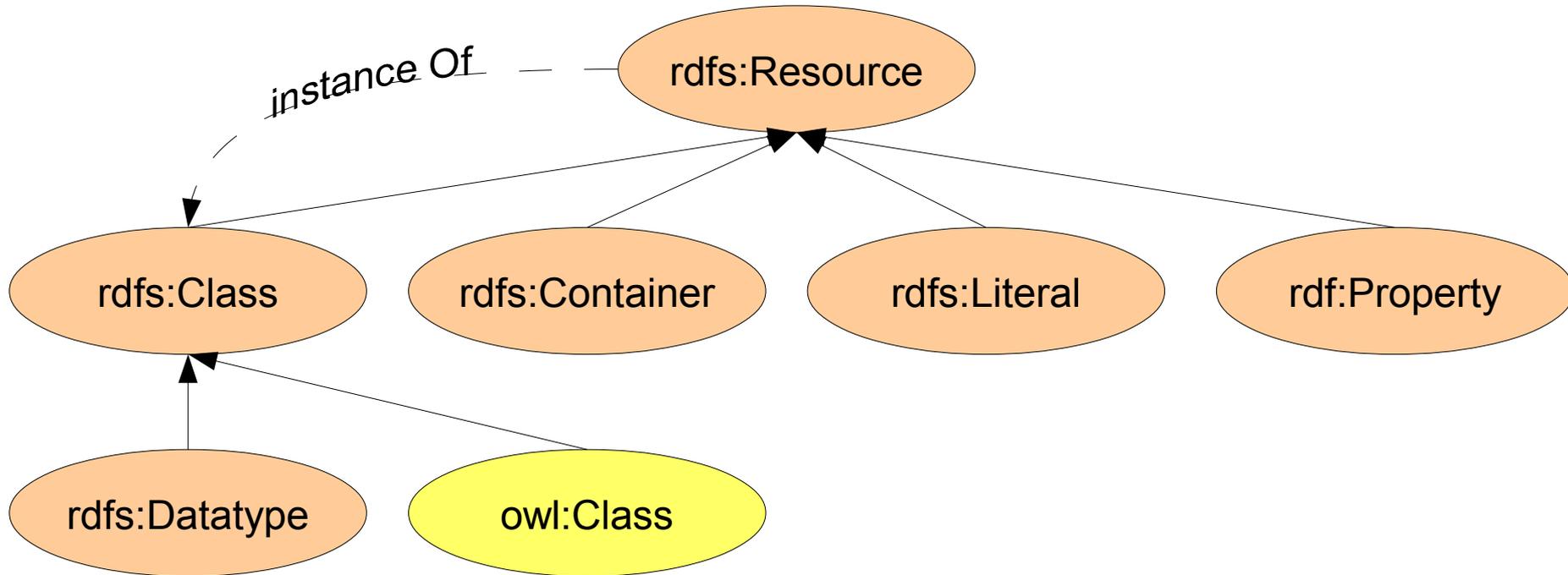
```
:TomsKlasse a rdfs:Class .  
:TomsKlasse foaf:name "Toms Klasse" .  
:Tom a :TomsKlasse .  
:Tom foaf:name "Tom" .
```

- Das darf man mit OWL-Klassen (in Lite/DL) nicht machen
- Sonst funktioniert das Reasoning nicht mehr
- `owl:Class` ist die Untermenge von `rdfs:Class`, die "sauber" verwendet wird

Vergleich OWL Lite/DL/Full

- OWL Lite:
 - Klassendefinition nur mit `subClassOf` und `intersectionOf`
 - Restriktion von Kardinalitäten nur mit 0 und 1
 - Qualitative Restriktion nicht mit `hasValue`
 - Explizite Typisierung
- OWL DL:
 - alle Sprachelemente erlaubt
 - saubere Trennung von Klassen, Instanzen, Object-/Data-/AnnotationProperties
 - Kardinalität nicht zusammen mit transitiven Properties verwendbar
 - Explizite Typisierung
- OWL Full:
 - alles ist erlaubt

Kompatibilität OWL mit RDFS



- Merke: nur OWL Full ist mit RDFS voll kompatibel!

Eine kleine Ontologie

- Domäne: Unversität
- Erster Schritt: Klassen definieren

```
:UniversityMember rdfs:subClassOf foaf:Person .  
:Student rdfs:subClassOf :UniversityMember .  
:UniversityEmployee rdfs:subClassOf :UniversityMember .  
:Assistant rdfs:subClassOf :UniversityEmployee .  
:Professor rdfs:subClassOf :UniversityEmployee .
```

```
:Lecture rdfs:subClassOf :Course .  
:Seminar rdfs:subClassOf :Course .
```

Eine kleine Ontologie

- Zweiter Schritt: Beziehungen zwischen Klassen definieren

```
:attends rdfs:domain :Student ;  
        rdfs:range :Course .  
:attendedBy owl:inversePropertyOf :attends .  
  
:teaches rdfs:domain :UniversityEmployee ;  
        rdfs:range :Course .  
:taughtBy owl:inversePropertyOf :teaches .
```

Eine kleine Ontologie

- Dritter Schritt: Weitere Konzepte und Axiome definieren

```
:SmallClass owl:equivalentClass [  
  a owl:Restriction ;  
  owl:onProperty :attendedBy ;  
  owl:maxCardinality 10^^xsd:integer .  
]
```

```
:LazyEmployee owl:equivalentClass [  
  a owl:Restriction ;  
  owl:onProperty :teaches ;  
  owl:allValuesFrom :Seminar .  
]
```

Schlussfolgern mit Ontologien

- Wann können wir folgern, dass Karl-Heinz ein fauler Dozent ist?
- Probieren wir's aus...

Komplexität von Ontologien



- Reasoning (Schlussfolgern) mit Ontologien ist oft wichtig
- ...und meist ein "teures" Unterfangen

- Die Performance von Reasonern hängt stark von der Komplexität der Ontologie ab
- Die meisten brauchbaren Ontologien sind OWL DL
 - aber hier gibt es deutliche Unterschiede

- Wie bestimmt man die Komplexität?
 - Komplexitätsklassen für Ontologien

Einfache Ontologien: ALC



- ALC: Attribute Language with Complement
- Erlaubte Konstrukte:
 - subClassOf, equivalentClass
 - unionOf, complementOf, disjointWith
 - Restriktionen: allValuesFrom, someValuesFrom
 - domain, range
 - sowie die Definition von Individuen

- Komplexere Ontologien werden mit Buchstabenkürzeln bezeichnet
- Dabei ist
 - S = ALC plus transitive Properties (die Basis)
 - H = Hierarchien von Properties (subPropertyOf)
 - O = abgeschlossene Klassen (oneOf)
 - I = Inverse Rollen (inversePropertyOf)
 - N = numerische Restriktionen (min/maxCardinality)
 - F = Funktionale Properties
 - Q = qualifizierende numerische Restriktionen (erst ab OWL2)
 - (D) = Verwendung von DatatypeProperties

Beispiel: Komplexität

- Unsere Beispielontologie hat die Komplexität SIN, da sie
 - inverse Properties und
 - Zahlenrestriktionenverwendet*
- Fügen wir noch eine abgeschlossene Klasse hinzu, haben wir SOIN.

*) die Konstrukte aus der importierten FOAF-Ontologie nicht berücksichtigt

Vorsicht, Falle!

- Auch bei numerischen Restriktionen tauchen Datenwerte auf:

```
:SmallClass owl:equivalentClass [  
  a owl:Restriction ;  
  owl:onProperty :attendedBy ;  
  owl:maxCardinality 10^^xsd:integer .  
] .
```

- Diese zählen allerdings nicht für (D)!

OWL2 – die nächste Generation

- Der erste OWL-Standard wurde 2004 verabschiedet
- OWL2: 2009
 - Syntaktische Vereinfachungen
 - Neue Sprachmittel
 - Profile

OWL2: Syntaktische Vereinfachungen



- Disjoint Classes und Disjoint Union

- OWL 1:

```
:Wine owl:equivalentClass [  
  a owl:Class ;  
  owl:unionOf (:RedWine :RoséWine :WhiteWine) ] .  
:RedWine owl:disjointWith :RoséWine, :WhiteWine .  
:RoséWine owl:disjointWith :WhiteWine .
```

- OWL 2:

```
:Wine owl:disjointUnionOf (:RedWine :RoséWine :WhiteWine ) .
```

- auch möglich:

```
_:x a owl:AllDisjointClasses ;  
  owl:members (:RedWine :RoséWine WhiteWine ) .
```

OWL2: Syntaktische Vereinfachungen

- Negative(Object|Data)PropertyAssertion
- ermöglichen negative Aussagen
- z.B.: Paul ist nicht der Vater von Peter

```
_x [ a owl:NegativeObjectPropertyAssertion;  
      owl:sourceIndividual :Paul ;  
      owl:targetIndividual :Peter ;  
      owl:assertionProperty :vaterVon ] .
```

- Warum ist das eine syntaktische Vereinfachung?
- Wie soll das denn sonst gehen?

OWL2: Neue Property-Typen

- Reflexive, irreflexive und asymmetrische Properties
- Jeder Mensch ist verwandt mit sich selbst:
`:relativeOf a owl:ReflexiveProperty .`
- Niemand kann Elternteil von sich selbst sein:
`:parentOf a owl:IrreflexiveProperty .`
- Wenn x größer ist als y, kann y nicht größer als x sein
`:tallerThan a owl:AsymmetricProperty .`

OWL2: Property Chains und disjunkte Properties

- Großeltern sind die Eltern der Eltern,
Onkel sind Brüder von Eltern

```
:grandparentOf owl:propertyChainAxiom (:parentOf :parentOf) .  
:uncleOf owl:propertyChainAxiom (:brotherOf :parentOf) .
```

- Niemand kann gleichzeitig Großelternteil und Elternteil von derselben Person sein

```
:parentOf owl:propertyDisjointWith :grandParentOf .
```

OWL2: Qualifizierte Restriktionen

- Rückblende OWL:
Kardinalitäts- und Wertrestriktion nicht kombinierbar!
- d.h.: "klassisches" OWL kann nur entweder all/someValuesFrom oder min/maxCardinality
- Ein belesener Mensch hat mindestens 1000 Bücher gelesen (Zeitungen und Zeitschriften zählen nicht)

```
:LiteratePerson rdfs:subClassOf [  
  a owl:Restriction ;  
  owl:onProperty :hasRead;  
  owl:minQualifiedCardinality "1000"^^xsd:integer ;  
  owl:onClass :Book ] .
```

Analog existieren auch
owl:maxQualifiedCardinality und
owl:qualifiedCardinality (exakt)

OWL2: Neue Klassenkonstruktionen

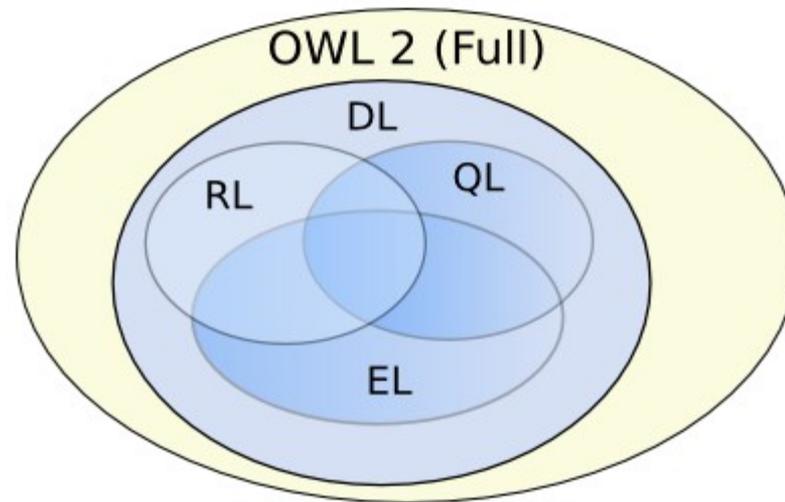


- Selbstrestriktion mit `hasSelf`
- Beispiel: Menge aller Autodidakten

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

OWL2 Profile

- Profile sind Untermengen von OWL2
 - EL, RL und QL
 - ähnlich wie Komplexitätsklassen
- Unterschiedliches Laufzeit- und Speicherverhalten
- Je nach Anforderungen



OWL2 Profile

- OWL2 EL (Expressive Language)
 - Schnelles Reasoning auf Standard-Ontologien
 - Einschränkungen z.B.
 - someValuesFrom, aber nicht allValuesFrom
 - keine Verwendung von inversen und symmetrischen Properties
 - kein unionOf und complementOf
- OWL2 QL (Query Language)
 - Schnelles Query Answering auf relationalen Datenbanken
 - Einschränkungen z.B.
 - kein unionOf, allValuesFrom, hasSelf, ...
 - keine Kardinalitäten und funktionale Properties

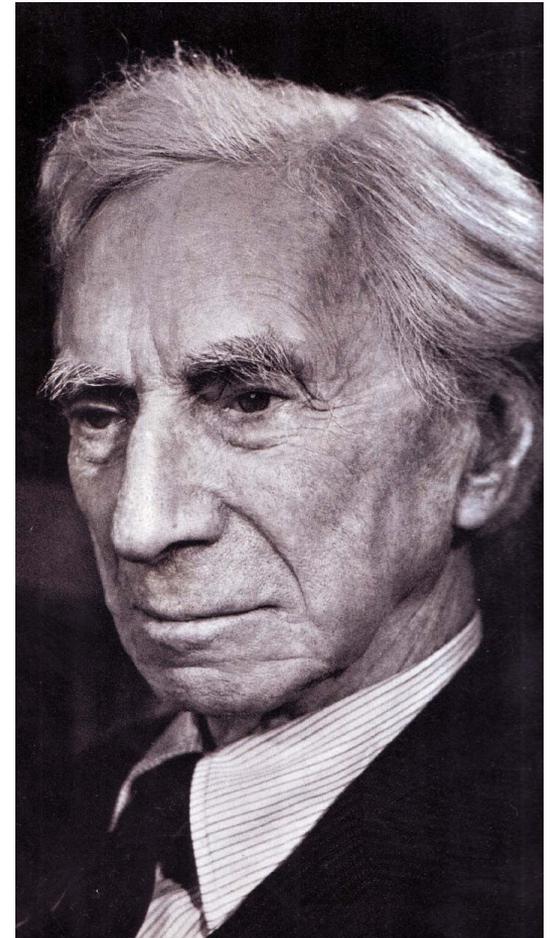
- OWL2 RL ("Rule Language")
 - Subset ähnlich wie Regelsprachen (z.B. Datalog)
 - subClassOf wird in Regel übersetzt (Head \leftarrow Body)
 - Head = Superklasse, Body = Subklasse
 - Einschränkungen z.B.
 - Nur qualifizierte Restriktionen mit 0 oder 1
 - Bestimmte Restriktionen für Head/Body
- Für alle Profile gilt
 - Reasoning kann in polynomieller Zeit implementiert werden
 - Reasoning auf der Vereinigung von zwei Profilen geht nur noch in exponentieller Zeit

Beispiel in OWL2: Das Barbier-Paradoxon



- Ein klasissches Paradoxon
(nach Bertrand Russell, 1918)
- In einer Stadt gibt es genau einen Barbier,
der genau all diejenigen rasiert, die sich
nicht selbst rasieren.

Wer rasiert den Barbier?



Beispiel in OWL2: Das Barbier-Paradoxon

▪ Definition der Klassen

```
:People owl:disjointUnionOf ( :PeopleWhoShaveThemselves  
:PeopleWhoDoNotShaveThemselves ) .
```

▪ Definition der Relationen

```
:shavedBy rdfs:domain :People .  
:shavedBy rdfs:range :People .  
:shaves owl:inverseOf :shavedBy .
```

▪ Jeder Mensch wird von genau einem Menschen rasiert

```
:People rdfs:subClassOf [  
  a owl:Restriction ;  
  owl:onProperty :shavedBy ;  
  owl:cardinality "1"^^xsd:integer ] .
```

Beispiel in OWL2: Das Barbier-Paradoxon



- Und dann brauchen wir noch genau einen Barbier:

```
:Barbers rdfs:subClassOf :People ;  
    owl:equivalentClass [  
        rdf:type owl:Class ;  
        owl:oneOf ( :theBarber )  
    ] .
```

Beispiel in OWL2: Das Barbier-Paradoxon



- Definition von Menschen, die sich selbst rasieren:

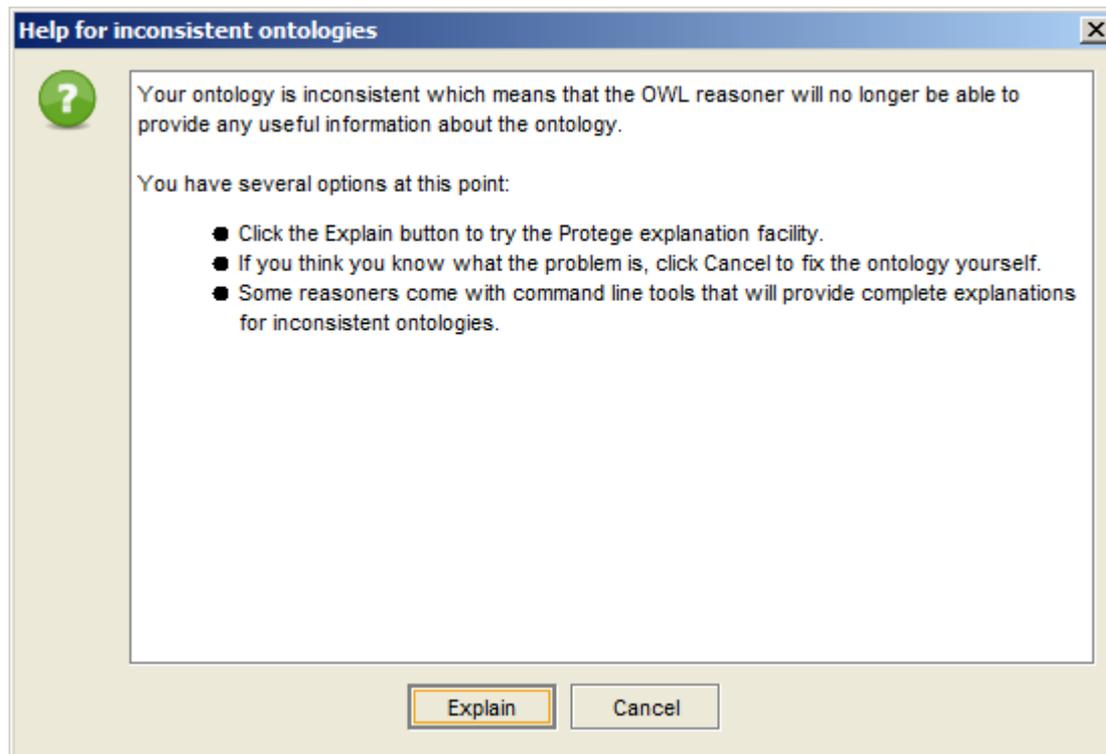
```
:PeopleWhoShaveThemselves owl:equivalentClass [  
  rdf:type owl:Class ;  
  owl:intersectionOf  
  ( :People  
    [  
      a owl:Restriction ;  
      owl:onProperty :shavedBy ;  
      owl:hasSelf "true"^^xsd:boolean  
    ]  
  )  
] .
```

Beispiel in OWL2: Das Barbier-Paradoxon

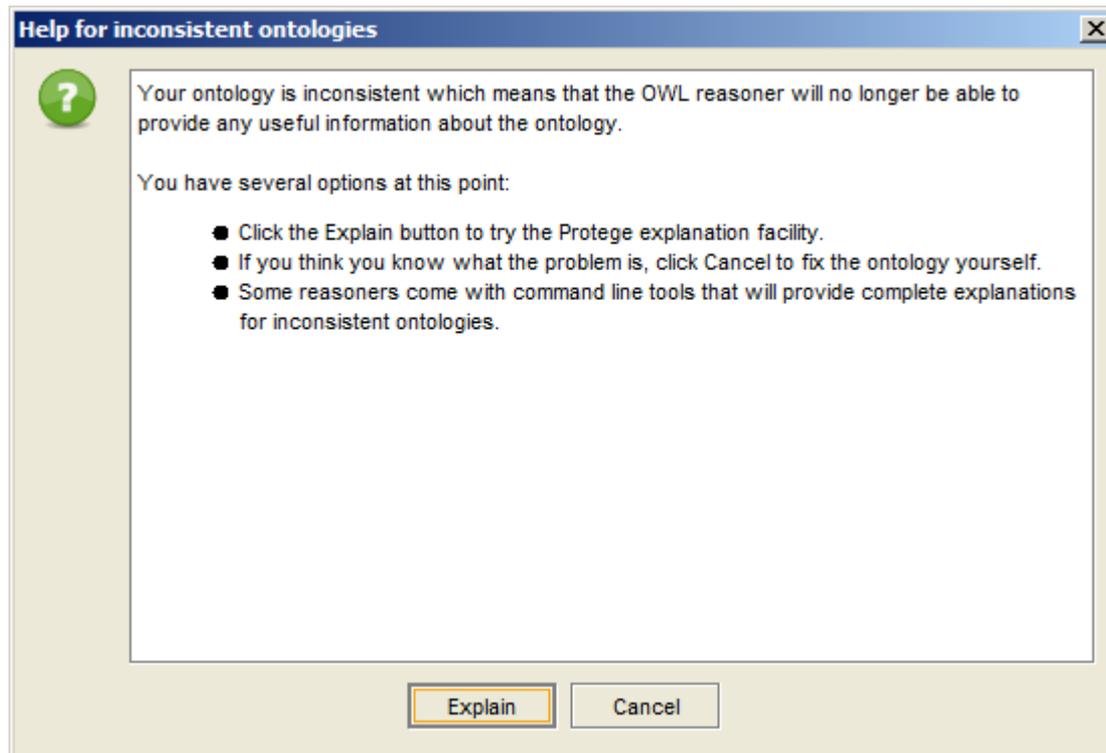
- Menschen, die sich nicht selbst rasieren, werden von Barbieren rasiert:

```
:PeopleWhoDoNotShaveThemselves owl:equivalentClass [  
  a owl:Class ;  
  owl:intersectionOf (  
    :People  
    [ a owl:Restriction  
      owl:onProperty :shavedBy ;  
      owl:allValuesFrom :Barbers  
    ]  
  )  
] .
```

Beispiel in OWL2: Das Barbier-Paradoxon



Beispiel in OWL2: Das Barbier-Paradoxon



Beispiel in OWL2: Das Barbier-Paradoxon



Experimental Protégé Explanation Heuristics

"Bad" Individuals Inconsistent Classes **Hot spots**

The axioms listed below are likely to be involved in an explanation of an inconsistency.
In particular, if the ontology contains no individuals then any explanation of why the ontology is inconsistent must include one of the axioms listed below.

Axioms

- **Barbers** EquivalentTo {theBarber}

Zusammenfassung

- OWL ist eine Sprache für Ontologien
- Mächtiger als RDF Schema
- Mengenorientiert
- Komplexe Zusammenhänge können formuliert werden

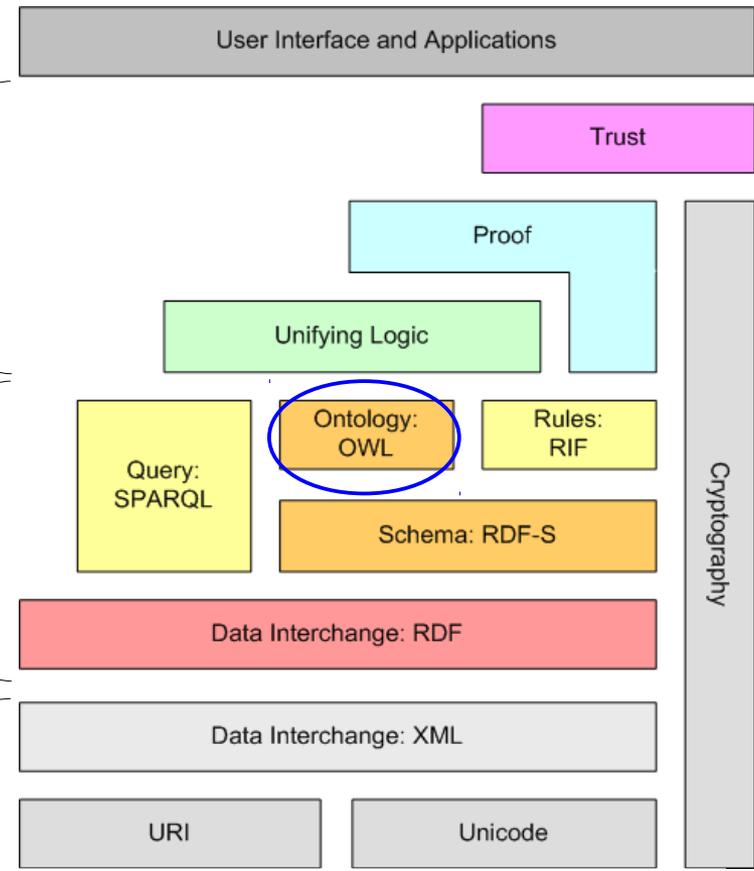
Semantic Web – Aufbau



here be dragons...

Semantic-Web-
Technologie
(Fokus der Vorlesung)

Technische
Grundlagen



Vorlesung Semantic Web



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Vorlesung im Wintersemester 2011/2012

Dr. Heiko Paulheim

Fachgebiet Knowledge Engineering