Vorlesung Semantic Web

TECHNISCHE UNIVERSITÄT DARMSTADT

Vorlesung im Wintersemester 2012/2013

Dr. Heiko Paulheim

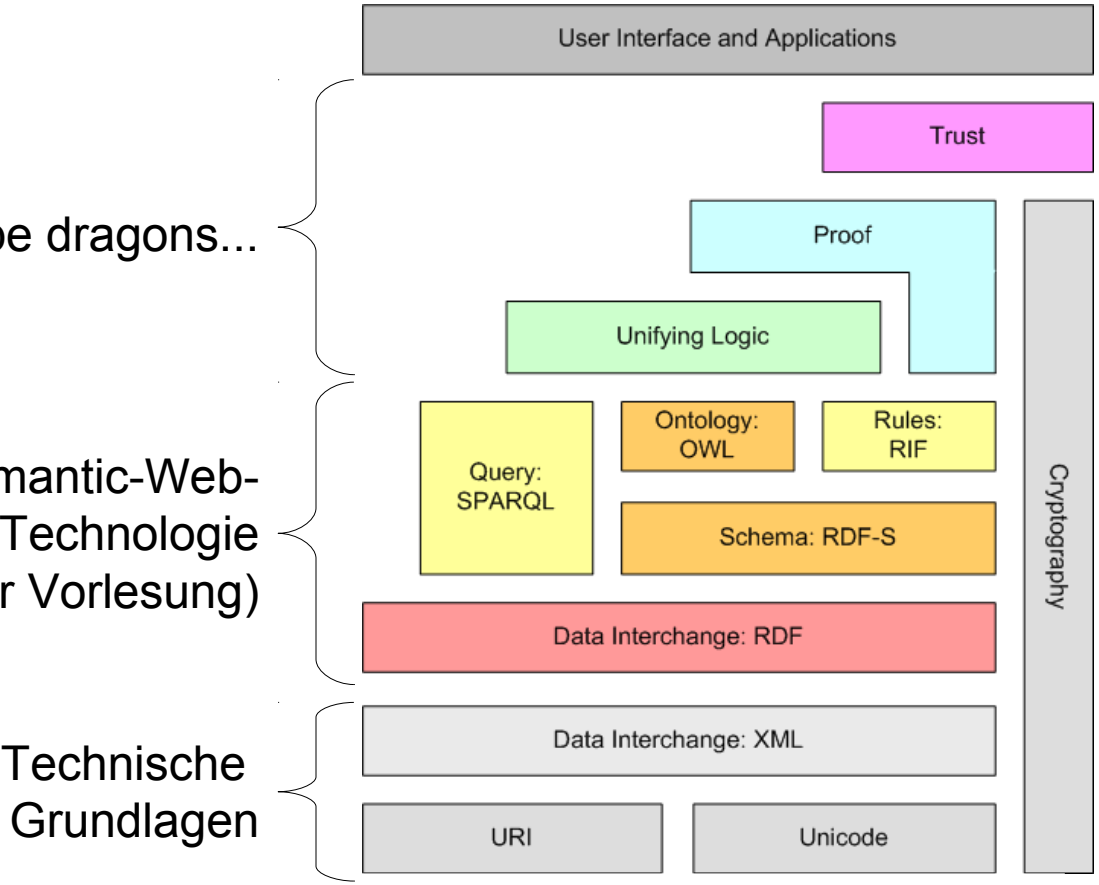
Fachgebiet Knowledge Engineering

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

TECHNISCHE UNIVERSITÄT DARMSTADT

- Der Semantic Web Layer Cake des W3C
  - bildet ein konzeptionelles Modell für das Semantic Web
  - ordnet die Standards des W3C an

- Es gibt aber noch eine Welt neben dem W3C...

Semantic Web Survey 2006/2007

TECHNISCHE UNIVERSITÄT DARMSTADT

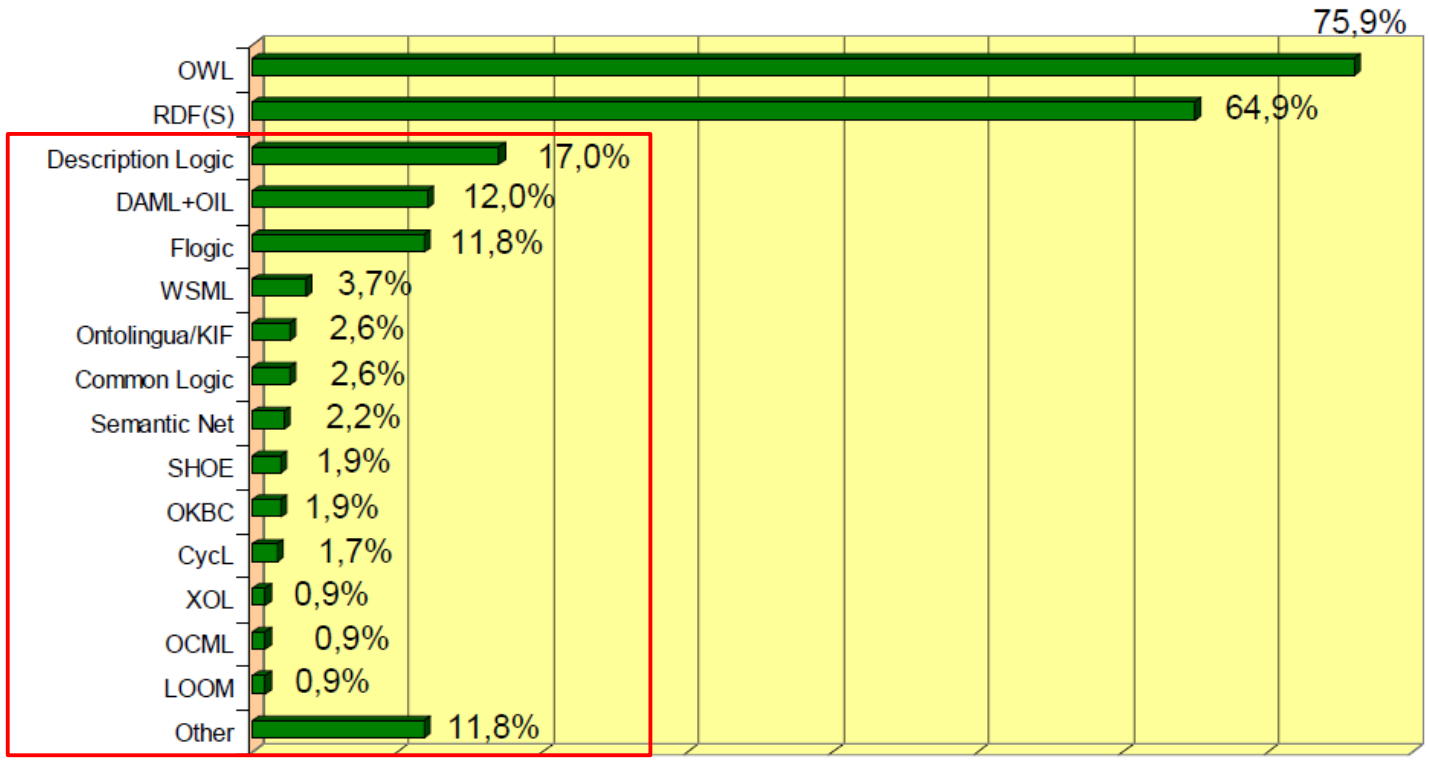
- Durchgeführt als Online-Umfrage
- über 600 Teilnehmer
- Fragen:
  - Welche Tools werden eingesetzt?
  - Welche Sprachen?
  - Welche Vorgehensmodelle?
  - ...

Semantic Web Survey 2006/2007

TECHNISCHE UNIVERSITÄT DARMSTADT

- Die Wahrheit über die W3C-Standards:

>50% Nicht-W3C-Standard-Sprachen!



Cardoso (2006): The Semantic Web Vision – Where are We?

Sprachen abseits der W3C-Standards

TECHNISCHE UNIVERSITÄT DARMSTADT

- Vorläufer und Verwandte von OWL & CO
  - DAML+OIL
  - SHOE
- Sprachen aus dem Bereich der Logik
  - insbesondere First Order Logic
  - Common Logic: ISO-Standard
  - Diverse Serialisierungen, z.B. KIF
- An Programmiersprachen angelehnte Ontologiesprachen
  - F-Logic
  - WSMO & Co

Vorläufer von OWL & Co.

TECHNISCHE UNIVERSITÄT DARMSTADT

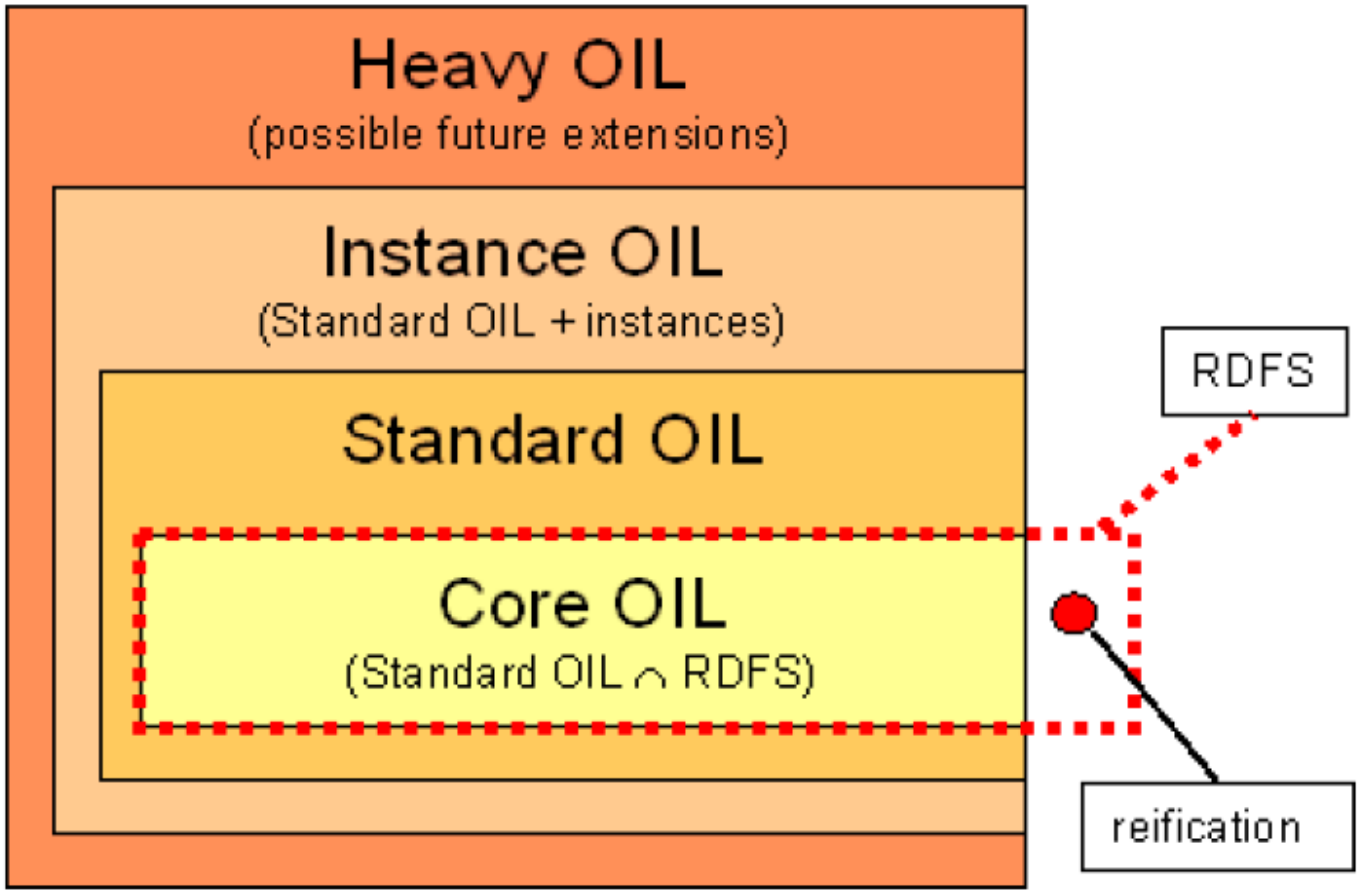
- Geschichte
  - 1999: erste Version von RDF standardisiert
  - 1999: erste Version von RDF Schema standardisiert
  - 2000: OIL als Erweiterung zu RDF Schema
  - 2000: DAML als Alternative zu RDF Schema
  - 2000/2001: DAML+OIL als Ontologiesprache
  - 2004: OWL als W3C-Standard
  - 2009: OWL2 als W3C-Standard

OIL

TECHNISCHE UNIVERSITÄT DARMSTADT

- Je nach Quelle
  - Ontology Inference Layer
  - Ontology Interchange Language
- Baut auf RDFS auf

Verhältnis von OIL und RDFS



Fensel et al. (2001): OIL: Ontology Infrastructure to Enable the Semantic Web

Verhältnis von OIL und RDFS

TECHNISCHE UNIVERSITÄT DARMSTADT

- Beispiel für OIL und RDFS
  - OIL wird in RDFS eingebettet
  - RDF Schema funktioniert auch ohne Erweiterung!

```
<rdfs:Class rdf:ID="Woman">
  <rdfs:subClassOf rdf:resource="#Person"/>
  <rdfs:subClassOf>
    <oil:NOT>
      <oil:hasOperand rdf:resource="#Man"/>
    </oil:NOT>
  </rdfs:subClassOf>
</rdfs:Class>
```

Mächtigkeit von OIL

TECHNISCHE UNIVERSITÄT DARMSTADT

- Ähnlich OWL
  - Mengenoperatoren AND, OR, NOT
    - enstpricht intersectionOf, unionOf, complementOf
  - Kardinalitäten
  - inverse, transitive, symmetrische und reflexive Properties

DAML

TECHNISCHE UNIVERSITÄT DARMSTADT

- DARPA Agent Markup Language
- Parallel zu OIL entwickelt
- ähnliche Ausdrucksmächtigkeit
  - aber keine Verbindung zu RDF Schema
  - minimale Unterschiede: z.B. keine lokalen Kardinalitätsrestriktionen in DAML

DAML+OIL

TECHNISCHE UNIVERSITÄT DARMSTADT

- Der Nachfolger von beiden Sprachen
  - gemeinsame Bezeichner
  - vereinheitlichtes Reasoning
- 2000 veröffentlicht
- Zweite Version 2001
  - Verwendet Datentypen von XML Schema
- direkter Vorläufer von OWL

Logiksprachen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Man unterscheidet
  - Prädikatenlogik erster Stufe
  - Prädikatenlogik höherer Stufe

- Es existieren Reasoner für solche Logiken
- ISO Common Logic Standard: Serialisierungen, z.B. XML

Prädikatenlogik erster Stufe: Inventar

TECHNISCHE UNIVERSITÄT DARMSTADT

- Prädikate: einstellige, mehrstellige
  - einstellige für Klassen
  - mehrstellige für Relationen
- Konjunktion, Disjunktion, Verneinung: $\wedge$, $\vee$, $\neg$
- Implikation, Äquivalenz, Gleichheit: $\rightarrow$, $\leftrightarrow$, $=$
- Quantoren: $\forall$, $\exists$
- Variablen und Konstanten
  - Konvention: Variablen beginnen mit Kleinbuchstaben, Konstanten mit Großbuchstaben

Ontologien in Prädikatenlogik

TECHNISCHE UNIVERSITÄT DARMSTADT

- Klassen und Subklassen
  - ∀x: Hund(x) → Säugetier(x)
- Vereinigungsmenge, Schnittmenge, Komplementmenge
  - ∀x: Kind(x) ∨ Erwachsener(x) ↔ Mensch(x)
  - ∀x: Student(x) ∧ Kind(x) ↔ MinderjährigerStudent(x)
  - ∀x: Kind(x) ↔ ¬Erwachsener(x)
- Abgeschlossene Mengen
  - ∀x: Kontinent(x) ↔ x=Afrika ∨ x=Asien ∨ x=Europa …

Ontologien in Prädikatenlogik

TECHNISCHE UNIVERSITÄT DARMSTADT

- Relationen, Domain und Range, Subrelationen
  - ∀x,y: verwandtMit(x,y) → Mensch(x)
  - ∀x,y: verwandtMit(x,y) → Mensch(y)
  - ∀x,y: vaterVon(x,y) → verwandtMit(x,y)
- Symmetrie, inverse Relationen, Transivititität, Reflexivität
  - ∀x,y: verwandtMit(x,y) → verwandtMit(y,x)
  - ∀x,y: kindVon(x,y) → elternteilVon(y,x)
  - ∀x,y,z: verwandtMit(x,y) ∧ verwandtMit(y,z) → verwandtMit(x,z)
  - ∀x: verwandtMit(x,x)
- Auch mehrwertige Relationen sind möglich
  - hatZutat(rezept,substanz,menge)

Ontologien in Prädikatenlogik

TECHNISCHE UNIVERSITÄT DARMSTADT

- Qualifizierende Restriktionen
  - ∀x: Mensch(x) → ∃y: vaterVon(y,x) ∧ Mensch(y)
    entspricht: owl:someValuesFrom
  - ∀x,y: BoyGroup(x) ∧ mitgliedVon(y,x) → Boy(y)
    entspricht owl:allValuesFrom
  - ∀x: EuropäischeStadt(x) → liegtIn(x,Europa)
    entspricht owl:hasValue

Ontologien in Prädikatenlogik

TECHNISCHE UNIVERSITÄT DARMSTADT

- Quantifizierende Restriktionen
  - minimale Kardinalität, z.B. 2:
  - $\forall x$: Mensch(x)
    $\rightarrow \exists y1,y2$: elternteilVon(y1,x) $\wedge$ elternteilVon(y2,x) $\wedge \neg(y1=y2)$
  - maximale Kardinalität, z.B. 2
  - $\forall x,y1,y2,y3$:
    Mensch(x) $\wedge$ elternteilVon(y1,x) $\wedge$ elternteilVon(y2,x)
    $\wedge$ elternteilVon(y3,x)
    $\rightarrow (y1=y2) \vee (y2=y3) \vee (y1=y3)$

Ontologien in Prädikatenlogik

TECHNISCHE UNIVERSITÄT DARMSTADT

- Beobachtung:
  - Alles, was in OWL DL möglich ist, geht auch in Prädikatenlogik
  - plus einiges mehr
- Beispiel aus SWRL-Vorlesung:
  - $\forall x,y$: kindVon(x,y) $\wedge$ Frau(x) $\rightarrow$ tochterVon(x,y)
- Quantoren verschachteln
  - Beispiel: Jeder Mensch kennt jemanden, der alle seine Verwandten kennt
  - $\forall x$: Mensch(x) $\rightarrow \exists y$: kennt(x,y) $\wedge$ ($\forall z$: verwandtMit(x,z) $\rightarrow$ kennt(y,z))
- Prädikatenlogik ist also weitaus mächtiger als OWL DL!

Common Logic

TECHNISCHE UNIVERSITÄT DARMSTADT

- ISO-Standard ISO/IEC 24707 (2007)
- Definiert
  - die Elemente von Prädikatenlogik, d.h., Terme, Variablen, etc.
  - verschiedene Serialisierungen
    - CLIF
    - XCL

CLIF und KIF

TECHNISCHE UNIVERSITÄT DARMSTADT

- Common Logic Interchange Format (Teil des ISO-Standards)
- Knowledge Interchange Format (90er-Jahre)
  - eine der ersten Ontologie-Sprachen
  - auch bekannt als *Ontolingua*
  - syntaktisch auf LISP basierend:

```
(<=>
  (instance ?PHYS Physical)
  (exists (?LOC ?TIME)
    (and
      (located ?PHYS ?LOC)
      (time ?PHYS ?TIME))))
```

- Definition von *Physical*: zu einer Zeit an einem Ort lokalisiert

XCL

TECHNISCHE UNIVERSITÄT DARMSTADT

- eXtended Common Logic Markup Language (Teil des ISO-Standards)

```
<forall>
  <var name="PHYS"/>
  <iff>
    <exists>
      <var name="LOC"/>
      <var name="TIME"/>
      <and>
        <atomic>
          <relation>
            <term name="located"/>
          </relation>
          <term name="PHYS"/>
          <term name="LOC"/>
        </atomic>
      </and>
    ...
```

Reasoning mit Common Logic

TECHNISCHE UNIVERSITÄT DARMSTADT

- Theorembeweiser:
  - Software, die Anfragen an Logik-Programme löst
  - und Begründungen (Beweise) mitliefert
  - Prinzipiell wie Reasoning für Ontologien

Higher Order Logic

TECHNISCHE UNIVERSITÄT DARMSTADT

- Bisher: Prädikatenlogik erster Ordnung (First Order Logic)
- Es gibt bestimmte Dinge, die in Prädikatenlogik erster Ordnung ausgeschlossen sind
  - aus gutem Grund
  - um noch sinnvolles Reasoning zu ermöglichen

- Parallele: Beschränkungen für OWL DL
  - Instanzen, Properties und Klassen müssen strikt getrennt sein

```
:Mensch a owl:Class .
:Mensch definedBy :MaxMustermann .
:MaxMustermann a :Mensch .
```

Higher Order Logic

TECHNISCHE UNIVERSITÄT DARMSTADT

- Beispiel: symmetrische Relation in Prädikatenlogik

  $\forall x,y$: kennt(x,y) $\leftrightarrow$ kennt(y,x)

- Wir könnten das ja auch so probieren:

  $\forall r$: symmetricRelation(r) $\leftrightarrow$ ($\forall x,y$: r(x,y) $\leftrightarrow$ r(y,x))
  symmetricRelation(kennt)
  kennt(Peter,Paul)

Higher Order Logic

TECHNISCHE UNIVERSITÄT DARMSTADT

- Higher Order Logic
  - erlaubt quantifizierende Aussagen über Prädikate
  - erlaubt Prädikaten, als Instanzen aufzutauchen

- Definition allgemeiner Relationen
  - Objekte, die mit irgend einer Relation verbunden sind

    $\forall x,y$: related(x,y) $\leftrightarrow \exists r$: r(x,y)

- Solche Konstrukte sind manchmal nützlich
  - machen aber das Reasoning sehr aufwändig

Beispiel: Higher Order Logic

TECHNISCHE UNIVERSITÄT DARMSTADT

- Kann man z.B. für temporales Reasoning verwenden:

  giltVonBis(bundesKanzlerVon(AngelaMerkel,BRD),2000,2013)

  giltVonBis(X,T1,T2) ∧ (T>T1) ∧ (T<T2) → giltZumZeitpunkt(X,T)

F-Logic: Grundlagen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Ontologiesprache basierend auf *Frames*
- Frame: Sammlung von Eigenschaften einer Klasse (Slots)
- Ähnlich: Klassenmodell, Datenbankmodell

Person	Mutter (Person)	Vater (Person)	Alter (int)
:Paul	:Martha	:Hans	24
:Martha	:Johanna	:Karl	47
...	...	...	...

F-Logic: Grundlagen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Erste Beobachtung:
  - Relationen sind an Klassen "gebunden"
  - und nicht "frei" wie in RDFS/OWL
- Vererbung
  - Relationen werden an Subklassen vererbt
  - Wertebereich und Kardinalität kann nicht weiter eingeschränkt werden
- Relationen
  - Daten-/Objektrelationen werden wie in OWL unterschieden
  - Hierarchien möglich
  - Symmetrie und Transitivität

F-Logic: Basis-Syntax

TECHNISCHE UNIVERSITÄT DARMSTADT

- Definition von Klassen und Subklassen

```
Author::Person .
```

- Definition von Instanzen

```
Shakespeare:Author .
```

F-Logic: Basis-Syntax

TECHNISCHE UNIVERSITÄT DARMSTADT

- Definition von Relationen

```
Author[hasWritten *=> Book] .
```

- Kardinalität von Relationen

```
Author[hasWritten{1:*} *=> Book] .
```

- Instanziierung von Relationen

```
Shakespeare[hasWritten -> Hamlet] .
```

F-Logic: Basis-Syntax

TECHNISCHE UNIVERSITÄT DARMSTADT

- Information über Instanzen kann auch kompakt dargestellt werden:

```
Shakespeare:Author[ born->1564,
                    hasWritten -> {Hamlet,Macbeth} ] .
```

- Ähnlich wie N3-Syntax

F-Logic: Basis-Syntax

TECHNISCHE UNIVERSITÄT DARMSTADT

- Information über Instanzen kann auch kompakt dargestellt werden:

```
Shakespeare:Author[ born->1564,
                    hasWritten -> {Hamlet,Macbeth} ] .
```

- Ähnlich wie N3-Syntax

F-Logic: Regeln

TECHNISCHE UNIVERSITÄT DARMSTADT

- Regeln sind wichtiger Mechanismus in F-Logic
  - fast alles wird in F-Logic darüber abgebildet

- z.B. PropertyChains

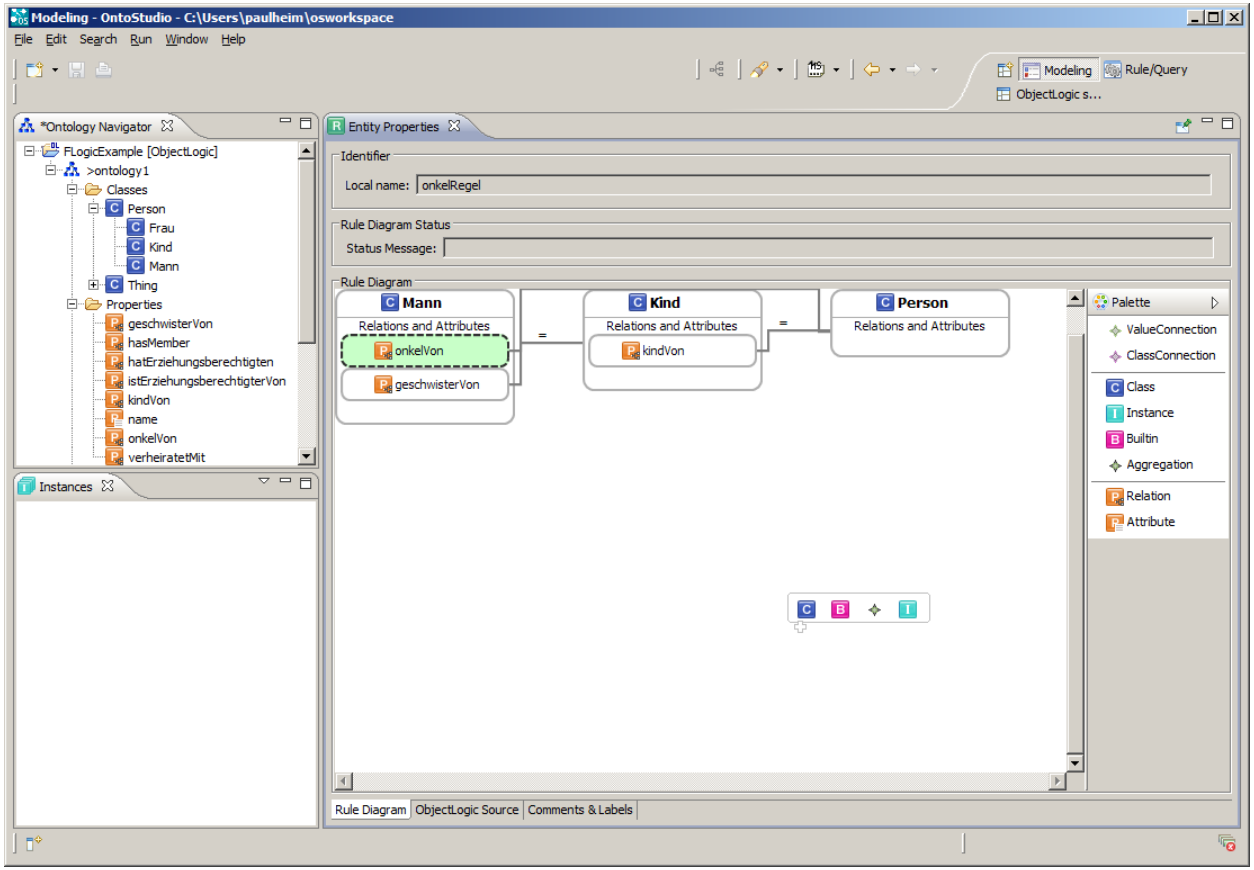
```
onkelVon(?X,?Z) :-
        ?X:Mann[geschwisterVon->?Y] and ?Z[kindVon->?Y] .
```

- Syntax: ähnlich wie Datalog
  - Head und Body getrennt durch `:-`
  - Variablen mit `?`

F-Logic: Regeln

TECHNISCHE UNIVERSITÄT DARMSTADT

- Graphische Regelbearbeitung, z.B. in OntoStudio



F-Logic: Regeln

TECHNISCHE UNIVERSITÄT DARMSTADT

- Quantoren in Regeln
  - Ein Autor ist eine Person, die mindestens ein Buch geschrieben hat

```
?X:Author :- ?X:Person
  AND (EXIST ?Y ?Y:Book and ?X[hatGeschrieben->?Y]).
```

  - Ein Nicht-Autor ist eine Person, die kein Buch geschrieben hat

```
?X:NonAuthor :- ?X:Person
  AND NOT(EXIST ?Y ?Y:Book and ?X[hatGeschrieben->?Y]).
```

  - Ein Starautor ist eine Person, die *nur* Bestseller geschrieben hat

```
?X[isStarAuthor->true] :- ?X:Person AND
  (FORALL ?Y (?X[hatGeschrieben->?Y] --> ?Y:Bestseller) ) .
```

F-Logic: Regeln

TECHNISCHE UNIVERSITÄT DARMSTADT

- Ein Starautor ist eine Person, die *nur* Bestseller geschrieben hat

```
?X[isStarAuthor->true] :- ?X:Person AND
  (FORALL ?Y (?X[hatGeschrieben->?Y] --> ?Y:Bestseller) ) .
```

- Ist diese Regel überhaupt korrekt?

- Die Bedingung ist auch erfüllt, wenn ?X gar keine Bücher geschrieben hat!

- Verbesserte Variante: zusätzlich auf Existenz prüfen

```
?X[isStarAuthor->true] :- ?X:Person AND
  (EXIST ?Z (?X[hatGeschrieben->?Z])) AND
  (FORALL ?Y (?X[hatGeschrieben->?Y] --> ?Y:Bestseller) ) .
```

F-Logic: Regeln

TECHNISCHE UNIVERSITÄT DARMSTADT

- Quantoren können geschachtelt werden
  - Ehrliche Menschen haben in all ihren Büchern jede Seite selbst geschrieben


```
?X[ehrlich->true] :- FORALL ?Y
  (?X[hatGeschrieben->?Y] -->
    (NOT EXIST ?S (?Y[hatSeite->?S] AND
      NOT(?X[hatGeschrieben->?S]) ))).
```

http://www.spiegel.de/spam/bild-746253-181841.html

F-Logic: Higher Order Logic

TECHNISCHE UNIVERSITÄT DARMSTADT

- F-Logic erlaubt Konstrukte aus Higher Order Logic
- Beschränkungen wie in OWL DL existieren nicht

- Beispiel:

  ?Z[?Y->?X] :- ?X[?Y->?Z] and ?Y:SymmetricProperty .

F-Logic: Abfragen

TECHNISCHE UNIVERSITÄT DARMSTADT

- F-Logic hat auch eine Abfragesprache integriert
  - Frage nach allen Autoren

```
?- ?X:Autor .
```

  - Frage nach allen Autoren und ihren Büchern

```
?- ?X:Autor[hatGeschrieben->?Y] .
```

- Alle gültigen Belegungen der enthaltenen Variablen werden ausgegeben
- Auch komplexere Ausdrücke sind möglich

```
?- ?X:Autor AND
   EXIST ?Y (?Y:Bestseller AND ?X[hatGeschrieben->?X]) .
```

F-Logic: Negation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Beobachtung: in F-Logic haben wir Negation zur Verfügung

```
?X[magNicht->?Y] :- not(?X[mag->?Y]) .
```

- Was wir bislang gelernt haben
  - Negation macht oft Schwierigkeiten
  - z.B. Unterscheidung OWL Lite/DL

F-Logic: Negation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Negation kann zu unerfreulichen Effekten führen

```
?X[magNicht->?Y] :- not(?X[mag->?Y] or ?X[istEgal->?Y])) .
?X[mag->?Y] :- ?X[kennt->?Y] and not(?X[magNicht->?Y]) .
```

- Gegeben folgende Abfrage:

```
?- ?X[mag->Stefan] .
```

- So könnte der Reasoner versuchen, die Abfrage abzuarbeiten

```
?X[mag->Stefan] .
?X[kennt->Stefan] and not(?X[magNicht->Stefan])) .
?X[kennt->Stefan] and not(not(?X[mag->Stefan] or
                              ?X[istEgal->?Y)) .
?X[kennt->Stefan] and not(not(?X[kennt->Stefan] and ...
```

Stratifikation

TECHNISCHE UNIVERSITÄT DARMSTADT

- F-Logic-Ontologien, die Negation enthalten, können unentscheidbar sein
  - Besonders problematisch: Folgerungszyklen, die Negationen enthalten
  - einfachster Fall:
    - `p(X) :- not(p(X)) .`
- Test: Stratifizierbarkeit

- Lat. *Stratum* (pl.: *Strata*): *Schicht*
  - Die F-Logic-Ontologie wird in Schichten aufgeteilt
  - Jedes Prädikat bekommt dabei eine Schicht zugeteilt
    - Recap: Klassenaxiome sind einstellige, Relationen zweistellige Prädikate

Stratifikation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Weise jedem Prädikat p eine Schicht S(p)
- Dabei müssen zwei Bedingungen erfüllt sein

- Für alle Regeln, die p im Kopf und ein unnegiertes Prädikat q im Körper haben
  - $S(q) \leq S(p)$

- Für alle Regeln, die p im Kopf und ein negiertes Prädikat q im Körper haben
  - $S(q) < S(p)$

Stratifikation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Betrachten wir dieses Beispiel

```
?X[magNicht->?Y] :- not(?X[mag->?Y]) .
?X[kennt->?Y] :- ?X[mag->?Y] .
```

- Es muss gelten:
  - S(mag) < S(magNicht)
  - S(mag) ≤ S(kennt)

- Solange keine weiteren Regeln auftreten, können wir z.B. wie folgt stratifizieren:
  - S(mag) = 0
  - S(magNicht) = 1
  - S(kennt) = 0

Stratifikation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Damit erhalten wir folgende Schichten:

```
?X[magNicht->?Y] :- not(?X[mag->?Y]) .
```

Schicht 1

```
?X[kennt->?Y] :- ?X[mag->?Y] .
```

Schicht 0

- Triviale Beobachtung:
  - bei Ontologien ohne Negation reicht immer eine Schicht!

Stratifikation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Betrachten wir jetzt noch mal dieses Beispiel

```
?X[magNicht->?Y] :- not(?X[mag->?Y]) .
?X[mag->?Y] :- ?X[kennt->?Y] and not(?X[magNicht->?Y]) .
```

- Wie lässt sich das stratifizieren?

- Wir bräuchten
  - S(mag) < S(magNicht)
  - S(magNicht) < S(mag)

- Das werden wir nicht finden!
  - Das Programm ist also nicht stratifizierbar

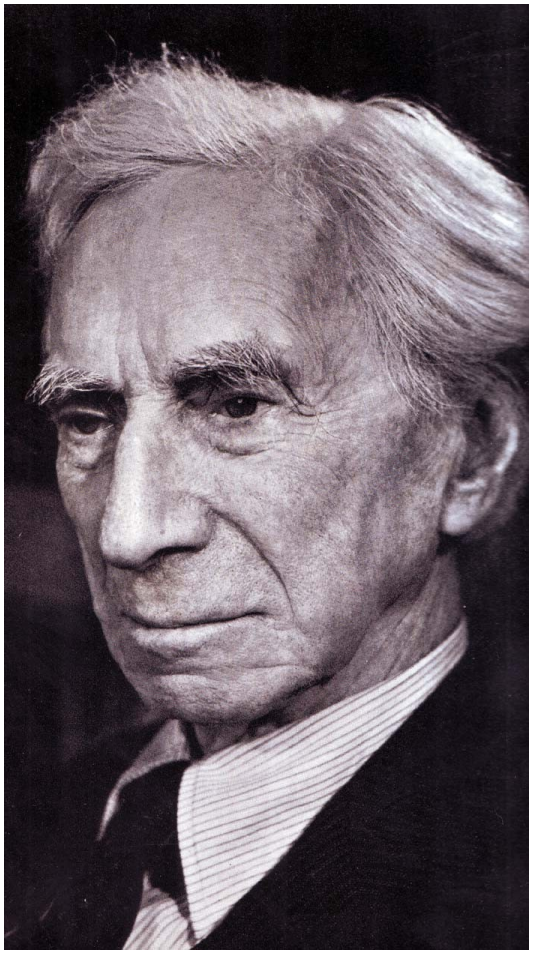
Wiederholung: Das Barbier-Paradoxon

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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.

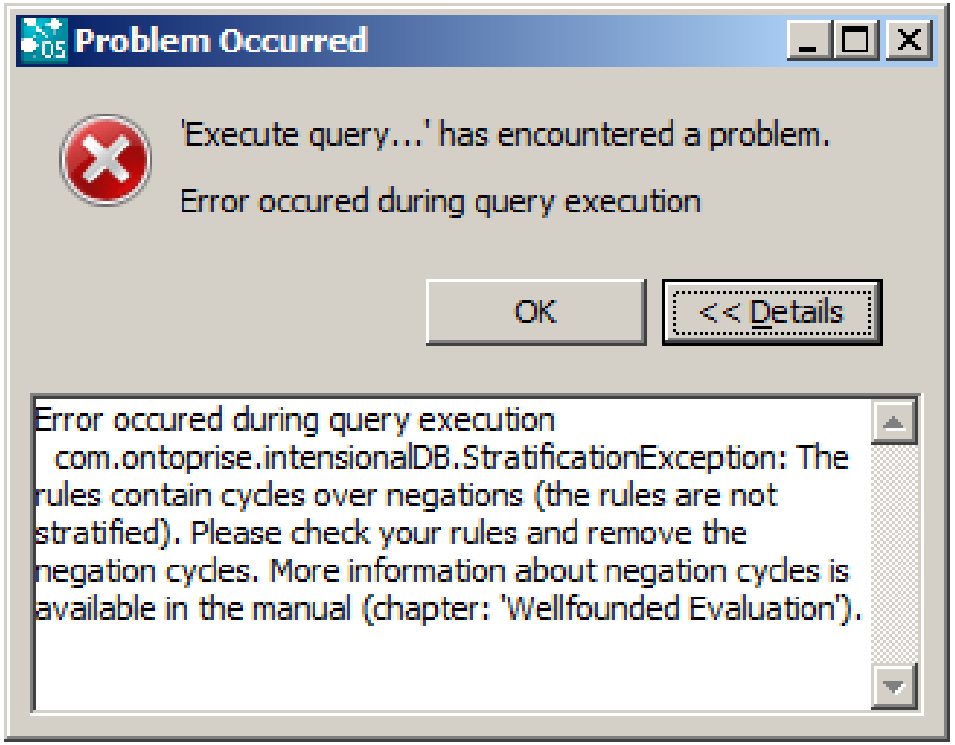
  Wer rasiert den Barbier?



Stratifikation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Barbier-Paradox in F-Logic:
  - theBarber[shaves->?X] :- not(?X[shaves->?X]) .
- Wir bräuchten also
  - S(shaves) < S(shaves)



F-Logic: Semantik

TECHNISCHE UNIVERSITÄT DARMSTADT

- F-Logic sieht oberflächlich ähnlich aus wie OWL-Ontologien
- Beispiel OWL:

```
:Mann rdfs:subClassOf :Mensch .
:Frau rdfs:subClassOf :Mensch .
:Hans a :Mann .
```

- Beispiel F-Logic:

```
Mann::Mensch .
Frau::Mensch .
Hans:Mann .
```

- Der Teufel liegt allerdings im Detail...

F-Logic: Semantik

TECHNISCHE UNIVERSITÄT DARMSTADT

- OWL
  - Open World Assumption
  - Non-unique Name Assumption
- F-Logic
  - Closed World Assumption
  - Unique Name Assumption

Semantik F-Logic vs. OWL

TECHNISCHE UNIVERSITÄT DARMSTADT

- Beispiel OWL:

```
:Mann rdfs:subClassOf :Mensch .
:Frau rdfs:subClassOf :Mensch .
:Hans a :Mann .
```

- Beispiel F-Logic:

```
Mann::Mensch .
Frau::Mensch .
Hans:Mann .
```

- Semantik OWL:
  - es gibt *mindestens* die (evtl. gleichen) Subklassen Mann und Frau
  - Hans könnte auch eine Frau sein
- Semantik F-Logic:
  - es gibt *genau* zwei verschiedene Subklassen von Mensch
  - Hans ist keine Frau

Recap: A Tale from the Road

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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

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

F-Logic: Semantik

TECHNISCHE UNIVERSITÄT DARMSTADT

- In F-Logic:

```
SingleAuthorWork::Work .
MultiAuthorWork::Work .
Work[hasAuthor {1:*} *=> Author].
```

- Definition von SingleAuthorWork:

```
?X:SingleAuthorWork :-
  ?X[hasAuthor->?Y] and
  not(exist ?Z (?X[hasAuthor->?Z] and ?Y!=?Z)).
```

closed world assumption

unique name assumption

F-Logic: Semantik

TECHNISCHE UNIVERSITÄT DARMSTADT

- Definition von MultiAuthorWork:

```
?X:MultiAuthorWork :- ?X:Work and not(?X:SingleAuthorWork).
```

- Das bedeutet:
  - alles, von dem wir nicht wissen, dass es SingleAuthorWork ist, ist automatisch MultiAuthorWork!
  - Closed World Assumption in Reinkultur

- Jetzt live in OntoStudio!

F-Logic: Instanzen erzeugen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Was F-Logic nicht unterstützt:
  - Zu jeder Person existiert ein Vater

```
exist ?V (?X[hatVater->?V]) :- ?X:Person .
```

- Quantoren im Kopf der Regel sind nicht erlaubt
  - Lösung: (kontrolliert) neue Instanzen erzeugen
  - Skolemisierung

Skolemisierung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Algorithmus:
  - nach Albert Thoralf Skolem (1887-1963)
  - Ersetze Variablen mit Existenzquantor durch ein neu gewähltes Funktionssymbol
  - Beispiel:

    $\forall$ x: Person(x) $\rightarrow \exists$ v: Mann(v) $\wedge$ hatVater(x,v) .

    wird zu

    $\forall$ x: Person(x) $\rightarrow$ Mann(f(x)) $\wedge$ hatVater(x,f(x)) .

- Beachte: alle über $\forall$ quantifizierten Variablen werden Argumente des neuen Funktionssymbols!

Skolemisierung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Das können wir auch direkt in F-Logic machen
  - Prädikate werden hier unterstützt

```
?X[hatVater->f(?X):Mann] :- ?X:Person .
```

- Was passiert hier? Gegeben:

```
Hans:Person .
```

- Dann schließt der Reasoner:

```
Hans[hatVater->f(Hans)] . f(Hans):Mann .
```

- Und auch

```
f(Hans)[hatVater->f(f(Hans))]. f(f(Hans)):Mann .
f(f(Hans))[hatVater->f(f(f(Hans)))]. f(f(f(Hans))):Mann .
...
```

Skolemisierung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Skolem-Terme können schnell zu einer Explosion der Instanzen führen
- Manche Reasoner können damit aber umgehen
  - Backward-Chaining
  - Es kommt darauf an, was man fragt

- Mögliche Abfrage:

```
?- Hans[hatVater->?X] .
```

- Unmögliche Abfrage:

```
?- ?X:Mann .
```

- Jetzt live in OntoStudio...

Skolemisierung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Merke:
  - Skolemform ist nur eine Näherung der ursprünglichen Formel!

- Angenommen, es wäre gegeben:

  Hans[hatVater->Peter] .
- Hier schließt der Reasoner zusätzlich:

  Hans[hatVater->f(Hans)] .

- Wegen Unique Name Assumption hat Hans jetzt zwei Väter...

Skolemisierung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Können wir das beheben?

```
?X[hatVater->f(?X):Mann] :- ?X:Person
  and not(exist ?V (?V:Mann and ?X[hatVater->?V])).
```

- Das würde unser Problem lösen!
- Ist aber nicht möglich...
  - warum?

- Achtung, Stratifizierung!
  - wir bräuchten S(hatVater) < S(hatVater) .
  - das ist also kein stratifizierbares Programm...

Skolemisierung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Weiteres Problem
  - es wird immer genau eine Instanz erzeugt
  - mit hatVater ist das auch unproblematisch

- Was aber ist mit

  $\forall$ x: Vater(x) $\rightarrow$ $\exists$ k: Person(k) $\wedge$ hatVater(k,x) .

- In F-Logic und Skolemform:

```
fk(?X):Person[hatVater->?X] :- ?X:Vater .
```

- Mit dieser Formel wird für jeden Vater *genau ein* Kind erzeugt
  - und wir haben Closed World Assumption!

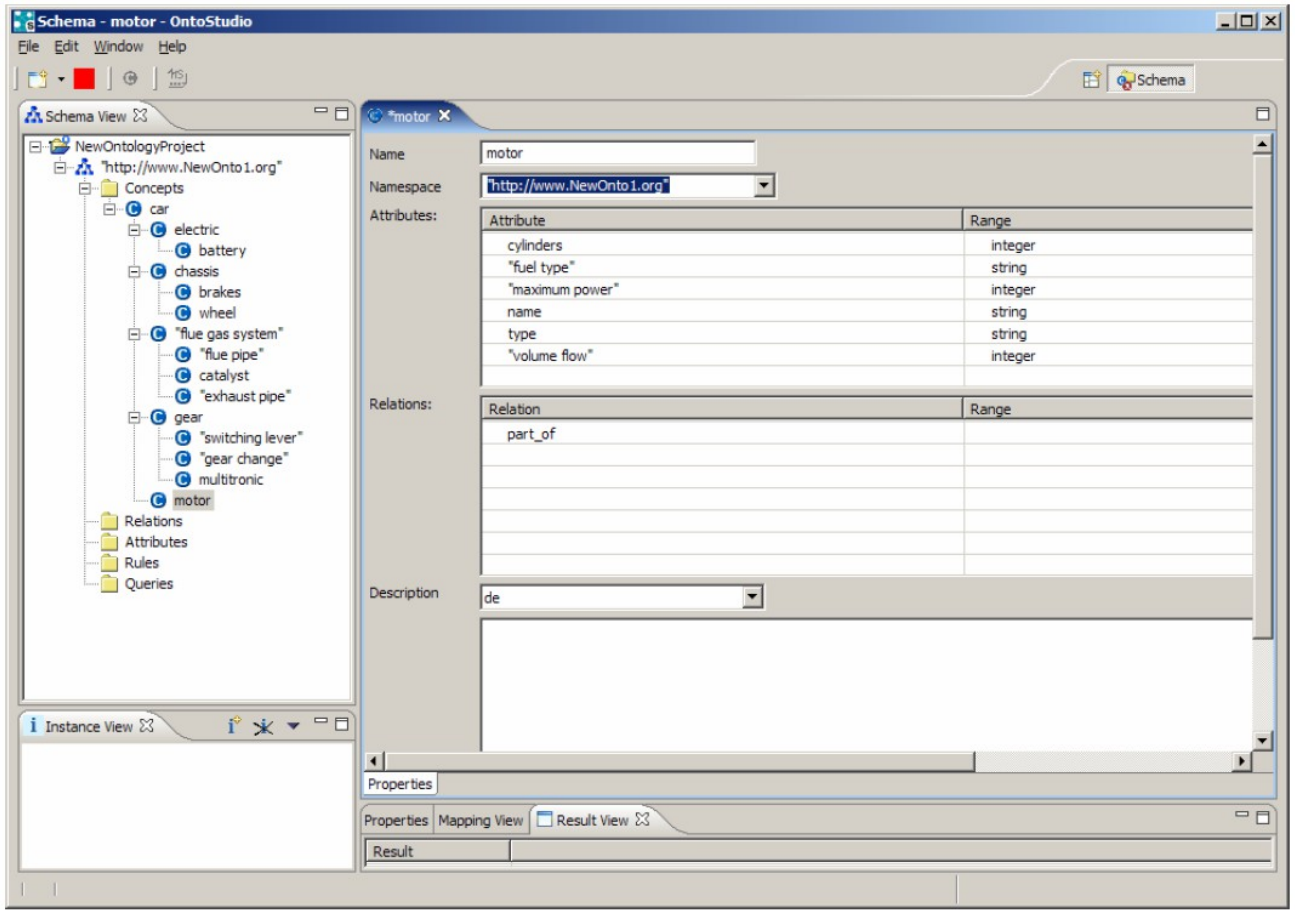
Anwendungsbeispiel: Expertensysteme für Automobile

TECHNISCHE UNIVERSITÄT DARMSTADT

- Konfigurationen überprüfen
  - Bestimmte Zusammenstellungen haben Probleme
  - diese kann man über Ontologiewissen erkennen
  - Komplexität schwer manuell handhabbar

- Domäne: Motoren, Bremsen, Katalysatoren
  - Der Filter im Katalysator muss zum Kraftstoff des Motors passen
  - Die Bremsen müssen ausreichend stark für die Motorstärke sein
  - ...

Anwendungsbeispiel: Expertensysteme für Automobile



Schnurr und Angele (2005): *Do Not Use This Gear with a Switching Lever! Automotive Industry Experience with Semantic Guides*

Anwendungsbeispiel: Expertensysteme für Automobile

TECHNISCHE UNIVERSITÄT DARMSTADT

- Mit Hilfe von Regeln können Fehlermeldungen erzeugt werden:

```
?C[hasError->"Mismatch of motor and filter"] :-
?C[hasMotor->?M] and ?M[usesFuel->?F]
and ?C[hasFilter->?F] and not (?F[supports->?M]) .
```

- Instanzdaten (Motorentypen, Filtertypen, technische Daten...)
  - liegen in eigenen Datenbanken
  - können mit speziellen Built-Ins herausgezogen werden
    - siehe Kapitel "Regeln"

F-Logic: Zusammenfassung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Umfassende Ontologiesprache
  - *eine* Sprache für Ontologie, Regeln, Abfragen
- Semantische Prinzipien
  - Closed World Assumption
  - Unique Name Assumption
- Unterschied zu RDF/OWL
  - Nicht nur syntaktischer Natur
  - Andere semantische Prinzipien
  - Anderes Reasoning

Vorlesung Semantic Web

TECHNISCHE UNIVERSITÄT DARMSTADT

Vorlesung im Wintersemester 2012/2013

Dr. Heiko Paulheim

Fachgebiet Knowledge Engineering