Lehrveranstaltung

Semantic Web Technologien

SS 2010

HTWG Konstanz

OWL

Web **O**ntology **L**anguage

- Begriff Ontologie
  - In der Philospophie:
    - Es gibt nur eine Ontologie (keine Ontologien)
    - Die “Lehre vom Sein”
    - Zu finden bei Aristoteles (Sokrates), Thomas von Aquin, Descartes, Kant, Hegel, Wittgenstein …
  - In der Informatik:

    Gruber (1993):
    “An Ontology is a **formal specification** of a **shared conceptualization** of a **domain** of interest”

    - Formelle Spezifikation → durch Maschinen interpretierbar
    - Shared → beruht auf einem Konsens
    - Conceptualization → beschreibt Begrifflichkeiten
    - Domain → bezieht sich auf ein “Thema” (Gegenstandsbereich)

OWL

- Anforderungen an eine Sprache zur Beschreibung von Ontologien:
  - Darstellung von Begriffen → Klassen
  - Instanzen von Klassen → Individuen
  - Begriffshierarchien (Taxonomien) → Vererbung
  - Beziehungen zwischen Individuen → Properties
  - Eigenschaften von Relationen (z.B Zielbereich, Transitivität, …)
  - Datentypen (z.B. Zahlen) → concrete domains
  - Logische Ausdrucksmittel
  - Klare Semantik

OWL

- RDF Schema als Ontologiesprache ?
  - Gut für einfache Ontologien, da begrenzte Semantik
  - Vorteil:
    - Schlussfolgerungen **schnell** und **effizient** durchführbar
  - Nachteil:
    - Zu wenig Ausdrucksstärke, komplexe Modellierung zu aufwändig / nicht durchführbar
  - Deswegen:
    - Rückgriff auf mächtigere Sprachen wie zum Beispiel
      - OWL
      - F-Logik
      - ...

OWL

- Wieso OWL und nicht WOL?
  Auszug aus einer Email von Tim Finin (27 Dec, 2001)

> Subject: Re: NAME: SWOL versus WOL
> ...
> I prefer the three letter WOL to the longer SWOL. How about OWL as a variation. The words would be the same (Ontology Web Language) but it has several advantages:
>
> (1) it has just one **obvious pronunciation** which is **easy on the ear**;
>
> (2) it opens up great opportunities for **logos**;
>
> (3) owls are associated with **wisdom**;
>
> (4) it has an interesting back **story**.
>
> OWL has probably been used for many computer languages and projects (see below), but I don't think that is a show stopper.
> ...

OWL

- OWL Web Ontology Language (1.0)
  - Offizielle W3C Recommendation vom 10. Februar 2004
    - Überarbeitung der Web-Ontologiesprache DAML+OIL
  - Seit 12.11.2009 ist die offizielle Recommendation zu OWL 2 veröffentlicht – dazu später mehr
  - Entworfen zur Verwendung bei Applikationen, die **Informationen verarbeiten** müssen anstatt sie nur Menschen zu präsentieren
  - OWL soll die **maschinelle Interpretierbarkeit** gegenüber XML, RDF und RDF(S) verbessern, indem sie **zusätzliches Vokabular** in Kombination mit einer **formalen Semantik** bietet.
  - OWL ist in drei Dialekte unterteilt:
    - OWL Lite
    - OWL DL
    - OWL Full

OWL

- Die drei OWL Dialekte
  - OWL Light
    - Einfach zu implementierende Untermenge von OWL DL
    - Dient zum Erschaffen einfacher Taxonomien und
    - Leicht axiomatische Ontologien
      - z.B. Nur Kardinalitäten von 0 und 1 möglich
    - Gedacht zur Migration von Thesauri und ähnlicher Taxonomien
    - Komplexität ExpTime im worst-case
  - OWL DL
    - DL → Description logic mit Mächtigkeit $\mathcal{SHOIN}(D)$
    - Berechenbar (alle Schlüsse können maschinell gezogen werden)
    - Entscheidbar (alle Berechnungen benötigen endlich viel Zeit)
    - Enhält alle Sprachkonstrukte von OWL
    - Einschränkungen für den Einsatz von RDF(S)
      - Eine Klasse darf nicht Instanz einer anderen Klasse sein, keine Reifikation ...
    - Wird von aktueller Software fast vollständig unterstützt
    - Komplexität NexpTime

OWL

- OWL Full
  - Maximale Ausdrückbarkeit
    - Ermöglicht prädikatenlogische Ausdrücke höheren Grades
  - Selbe Konstrukte wie DL
  - Keine Einschränkungen
    - Eine Klasse kann sowohl eine Kategorie von Individuen darstellen als auch selbst als Individuum existieren
  - → Ontologien sind unentscheidbar
  - Von Software nur bedingt unterstützt
- Jeder der Dialekte ist eine Erweiterung seines einfacheren Vorgängers
  - Jede gültige OWL Lite Ontologie ist gültiges OWL DL
  - Jede gültige OWL DL Ontologie ist gültiges OWL Full
  - Jeder Schluss, der in OWL Lite gilt, gilt auch in OWL DL
  - Jeder Schluss, der in OWL DL gilt, gilt auch in OWL Full

- Welchen Dialekt soll ich nun verwenden?
  - Abhängig von den Anforderungen
  - Wahl zwischen Lite und DL hängt von der erforderlichen Ausdrucksstärke ab
  - Wahl zwischen DL und Full hängt vor allem davon ab, inwiefern man die Metamodellierungsfähigkeiten von RDF(S) benötigt
    - Klassen von Klassen …
  - Weiterere wichtige Einschränkung bei Full:
    - Durch fehlende Entscheidbarkeit wird das Ziehen von Schlüssen schwierig bis unmöglich

OWL

- Kombatibilität mit RDF(S)
  - OWL Full ist eine Erweiterung von RDF
  - OWL Lite und DL sind Erweiterungen einer eingeschränkten Sicht auf RDF
  - Jedes gültige OWL Dokument ist ein gültiges RDF Dokument
  - Jedes RDF-Dokument ist gültiges OWL Full
    - Nur manche sind gültiges DL oder gar Lite
  - Vorsicht bei der Migration von RDF Dokumenten nach Lite/DL
    - Bei jeder URI, die als Klasse verwendet wird, muss explizit sichergestellt sein, dass sie vom Typ owl:Class ist (ähnliches gilt für Properties)
    - Bei jedem Individuum muss sichergestellt werden, dass es zu mindestens einer Klasse gehört (und wenn es nur owl:Thing ist)
    - Die URIs, die für Klassen, Properties und Individuen verwendet werden, müssen zueinander verschieden sein

- OWL Dokumente
  - Sind RDF Dokumente (zumindest in der Standard-Syntax)
  - Bestehen aus
    - Kopf mit allgemeinen Angaben zur Ontologie
      - v.a. Meta-Informationen
    - Rest mit der eigentlichen Ontologie
      - Klassen
      - Properties
      - Individuen
      - ...

OWL

- Beispiel:

```
<rdf:RDF xmlns="http://www.example.org/exampleOntology"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:owl="http://www.w3.org/2002/07/owl#" >

  <owl:Ontology rdf:about="">

    <rdfs:comment rdf:datatype="&xsd;string">
       This is an example Ontology providing classes and properties for …
    </rdfs:comment>

    <owl:versionInfo>v0.9</owl:versionInfo>

    <owl:imports rdf:resource="http://www.example.org/foo" />

    <owl:priorVersion rdf:resource="http://www.example.org/oldExampleOnt" />

  </owl:Ontology>

  <!--// Hier kommt der REST //-->

</rdf:RDF>
```

OWL

- Properties zur Beschreibung von OWL-Ontologien
  - Von RDF-S geerbt:
    - rdfs:comment, rdfs:label, rdfs:seeAlso, rdfs:isDefinedBy
  - Für die Versionierung
    - owl:versionInfo
    - owl:priorVersion
    - owl:backwardCompatibleWith
    - owl:incompatibleWith
    - owl:DeprecatedClass
    - owl:DeprecatedProperty
  - Desweiteren
    - owl:imports

Bestandteile von OWL Ontologien

OWL

- Die drei Bestandteile von Ontologieaxiomen
- Klassen
  - Vergleichbar mit den Klassen in RDFS
- Individuen
  - Vergleichbar mit Objekten in RDFS
- Rollen
  - Vergleichbar mit Properties in RDFS

OWL – Klassen

- Defintion von Klassen in OWL

```
<owl:Class rdf:ID="Student" />
```

- Vordefinierte Klassen:
  - owl:Thing → Oberklasse aller Klassen in OWL
  - owl:Nothing → Gegenteil von owl:Thing, enthält kein Individuum

OWL – Individuen

- Definition von Individuen in OWL
  - Geschieht durch Zuweisung einer owl:Class als Typ

```
<rdf:Description rdf:ID="MaxMustermann">
   <rdf:type rdf:resource="#Student" />
</rdf:Description>
```

- Kann auch kürzer geschrieben werden:

```
<Student rdf:ID="MaxMustermann" />
```

OWL – Abstrakte Rollen

- Definition abstrakter Rollen

```
<owl:ObjectProperty rdf:ID="Zugehoerigkeit" />
```

- Domain und Range abstrakter Rollen

```
<owl:ObjectProperty rdf:ID="Zugehoerigkeit" >
  <rdfs:domain rdf:resource="#Person" />
  <rdfs:range rdf:resource="#Organisation" />
</owl:ObjectProperty>
```

OWL – Konkrete Rollen

- Definition konkreter Rollen

```
<owl:DatatypeProperty rdf:ID="Matrikelnummer" />
```

- Konkrete Rollen haben einen Datentyp als Range

```
<owl:DatatypeProperty rdf:ID="Matrikelnummer">
  <rdfs:domain rdf:resource="#Student" />
  <rdfs:range rdf:resource="&xsd;string" />
</owl:DatatypeProperty>
```

- Viele XML-Datentypen können verwendet werden
  - Standard schreibt mindestens xsd:integer und xsd:string vor

```
<Student rdf:ID="MaxMustermann">
   <Zugehoerigkeit rdf:resource="#HTWG-Inf" />
   <Zugehoerigkeit rdf:resource="#ExampleGmbH" />
   <Matrikelnummer rdf:datatype="&xsd;string
                   value="A523234" />
</Student>
```

- Rollen sind im Allgemeinen nicht funktional
  - #HTWG-Inf und #ExampleGmbh können also durchaus zwei verschiedene Individuen bezeichnen

Beziehungen zwischen Klassen und Individuen

OWL

- Einfache Beziehungen zwischen Klassen

```
<owl:Class rdf:ID="Student">
   <rdfs:subClassOf="#HochschulAngehoeriger" />
</owl:Class>

<owl:Class rdf:ID="HochschulAngehoeriger" >
   <rdfs:subClassOf="#Person" />
</owl:Class>
```

- Durch Inferenz kann geschlossen werden, dass jeder Student eine Person ist

OWL

- Einfache Beziehungen zwischen Klassen

```
<owl:Class rdf:ID="Student">
  <rdfs:subClassOf="#HochschulAngehoeriger" />
</owl:Class>

<owl:Class rdf:ID="Buch" >
  <rdfs:subClassOf="#Publikation" />
</owl:Class>

<owl:Class rdf:ID="HochschulAngehoeriger">
  <owl:disjointWith rdf:resource="#Publikation"/>
</owl:Class>
```

- Durch Inferenz folgt, dass Student und Buch ebenfalls disjunkt sind (keine gemeinsamen Individuen besitzen)

OWL

- Einfache Beziehungen zwischen Klassen

```
<owl:Class rdf:ID="Buch">
   <rdfs:subClassOf="#Publikation" />
</owl:Class>

<owl:Class rdf:ID="Publikation" >
   <owl:equivalentClass="#Publication" />
</owl:Class>
```

- Durch Inferenz folgt, dass Buch eine Unterklasse von Publication ist.

OWL

- Einfache Beziehungen zwischen Klassen und Individuen

```
<Buch rdf:ID="AngelsAndDemons" >
  <Autor rdf:resource="#DanBrown" />
</Buch>

<owl:Class rdf:ID="Buch">
   <rdfs:subClassOf="#Publikation" />
</owl:Class>
```

- Durch Inferenz folgt, dass AngelsAndDemons eine Publikation ist

OWL

- Gleichheit von Individuen

```
<Professor rdf:ID="OliverEck">
<rdf:Description rdf:about="OliverEck">
   <owl:sameAs rdf:resource="ProfessorEck" />
</rdf:Description>
```

- Durch Inferenz folgt, dass "ProfessorEck" vom Typ Professor ist

- Verschiedenheit von Individuen kann mit
  owl:differentFrom
  ausgedrückt werden

OWL

- Unterschiedlichkeit von Individuen

```
<owl:AllDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
    <Person rdf:resource="MaxMustermann" />
    <Person rdf:resource="OliverEck" />
    <Person rdf:resource="GerdaMueller" />
    ...
  </owl:distinctMembers>
</owl:AllDifferent>
```

- Abkürzende Schreibweise an Stelle mehrerer owl:differentFrom Aussagen
- owl:AllDifferent und owl:distinctMembers werden ausschließlich in diesem Zusammenhang verwendet

OWL

- Abgeschlossene Klassen

```
<owl:Class rdf:ID="SekretaerinnenFBInf">
  <owl:oneOf>
    <Person rdf:resource="MichaelaBaier" />
    <Person rdf:resource="RoswithaFriedrich" />
  </owl:oneOf>
</owl:Class>
```

- Besagt, dass es nur genau diese beiden Personen als "Sekretärinnen im Fachbereich Informatik" gibt und keine weiteren.

Komplexe Klassen

OWL

- Logische Klassenkonstruktoren
  - Logisches UND (Konjunktion)
    `owl:intersectionOf`
  - Logisches ODER (Disjunktion)
    `owl:unionOf`
  - Logisches NICHT (Negation)
    `owl:complementOf`
- Mit Hilfe dieser Konstruktoren können aus einfachen Klassen komplexe Klassen erzeugt werden

OWL

```
<owl:Class rdf:ID="SekretaerinnenFBInf">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Sekretaerinnen" />
        <owl:Class rdf:about="#AngehoerigeFBInf" />
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
```

- Durch Inferenz folgt zum Beispiel, dass alle Sekretärinnen des Fachbereichs Informatik Sekretärinnen und Angehörige des Fachbereich Informatik sind.

OWL

```
<owl:Class rdf:ID="AngehoerigeFBInf">
  <owl:equivalentClass>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#AngestellteFBInf" />
        <owl:Class rdf:about="#StudentenFBInf" />
      </owl:unionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
```

- Durch Inferenz folgt zum Beispiel, dass jeder Student im Fachbereich Informatik ein Angehöriger des Fachbereichs ist.

OWL

- Rolleneinschränkung mit allValuesFrom
- Definition komplexer Klassen durch Rollen

```
<owl:Class rdf:ID="Kind">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hatVater" />
      <owl:allValuesFrom rdf:resource="#Mann" />
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

- Bedeutet, dass jeder Vater eines Kindes ein Mann sein muss.

OWL

- Rolleneinschränkung mit someValuesFrom

```
<owl:Class rdf:ID="KaesigePizza">
 <rdfs:subClassOf>
   <owl:Restriction>
     <owl:onProperty rdf:resource="#hatBelag" />
     <owl:someValuesFrom rdf:resource="#KaeseSorte" />
   </owl:Restriction>
 </rdfs:subClassOf>
</owl:Class>
```

- Bedeutet, dass jede “Käsige Pizza” mit mindestens einer Käsesorte belegt sein muss

OWL

- Rolleneinschränkung mit Maximalen Kardinalitäten

```
<owl:Class rdf:ID="ParabelZweiterOrdnung">
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:onProperty rdf:resource="#schneidetXAmPunkt" />
   <owl:maxCardinality
       rdf:dataType="&xsd;nonNegativeInteger" />
       2
   </owl:maxCardinality>
  </owl:Restriction>
 </rdfs:subClassOf>
</owl:Class>
```

- Bedeutet, dass eine Parabel zweiter Ordnung maximal 2 Schnittpunkte mit der x-Achse besitzen kann.

OWL

- Rolleneinschränkung mit Minimalen Kardinalitäten

```
<owl:Class rdf:ID="Vorlesung">
 <rdfs:subClassOf>
   <owl:Restriction>
     <owl:onProperty rdf:resource="#hatTeilnehmer" />
     <owl:minCardinality
          rdf:dataType="&xsd;nonNegativeInteger" />
       5
     </owl:minCardinality>
   </owl:Restriction>
 </rdfs:subClassOf>
</owl:Class>
```

- Bedeutet, dass jede Vorlesung mindestens 5 Teilnehmer haben muss

- Rolleneinschränkung mit genauen Kardinalitäten

```
<owl:Class rdf:ID="Kind">
 <rdfs:subClassOf>
   <owl:Restriction>
     <owl:onProperty rdf:resource="#hatGroßmutter" />
     <owl:cardinality
          rdf:dataType="&xsd;nonNegativeInteger" />
       2
     </owl:maxCardinality>
   </owl:Restriction>
 </rdfs:subClassOf>
</owl:Class>
```

- Bedeutet, dass jedes Kind genau zwei Großmütter hat

OWL

- Rolleneinschränkung mit hasValue

```
<owl:Class rdf:ID="EssenBeiOmaMustermann">
 <owl:equivalentClass>
   <owl:Restriction>
     <owl:onProperty rdf:resource="#gekochtVon" />
     <owl:hasValue rdf:resource="#OmaMustermann" />
   </owl:Restriction>
 </owl:equivalentClass>
</owl:Class>
```

- Bedeutet, dass jedes Essen, das bei Oma Mustermann stattfindet, auch von Oma Mustermann gekocht wird.
- owl:hasValue bezieht sich immer auf eine konkrete Instanz!

OWL

- Folgendes Beispiel ist equivalent zur letzten Folie

```
<owl:Class rdf:ID="EssenBeiOmaMustermann">
 <owl:equivalentClass>
   <owl:Restriction>
     <owl:onProperty rdf:resource="#gekochtVon" />
     <owl:someValuesFrom>
       <owl:oneOf rdf:parseType="Collection">
         <owl:Thing rdf:about="#OmaMustermann" />
       </owl:oneOf>
     </owl:someValuesFrom>
   </owl:Restriction>
 </owl:equivalentClass>
</owl:Class>
```

Rolleneigenschaften

OWL

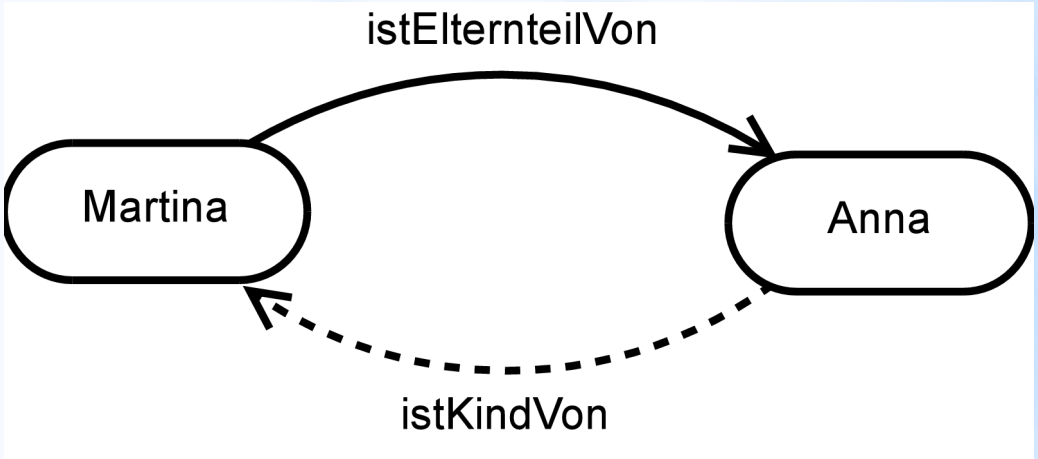
```
<owl:ObjectProperty rdf:ID="istBefreundetMit">
   <rdfs:subPropertyOf rdf:resource="kennt" />
</owl:ObjectProperty>
```

- Ebenso wie bei Klassen gibt es owl:equivalentProperty
- Rollen können aber zum Beispiel auch invers zueinander sein:

```
<owl:ObjectProperty rdf:ID="">
   <owl:inverseOf rdf:resource="" />
</owl:ObjectProperty>
```

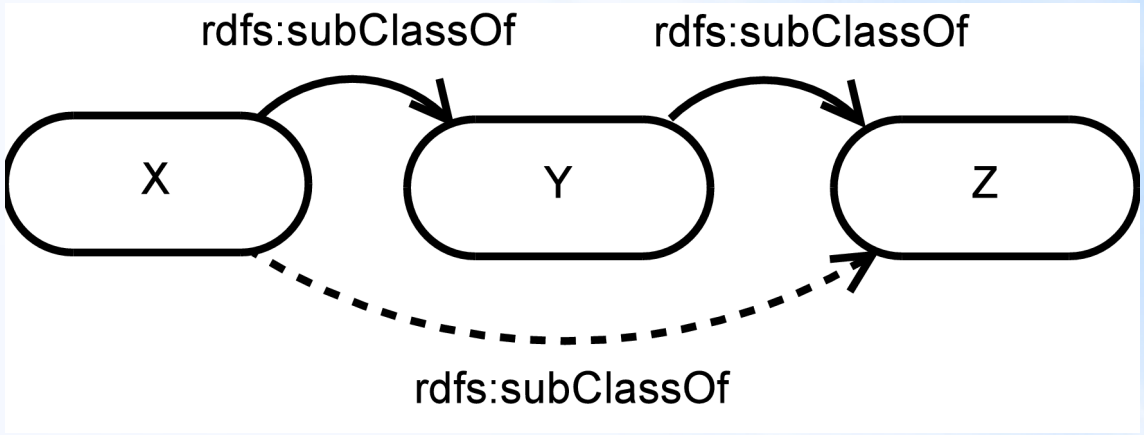
OWL

- Inverse Properties (Kennzeichnung mit **owl:inverseOf**)



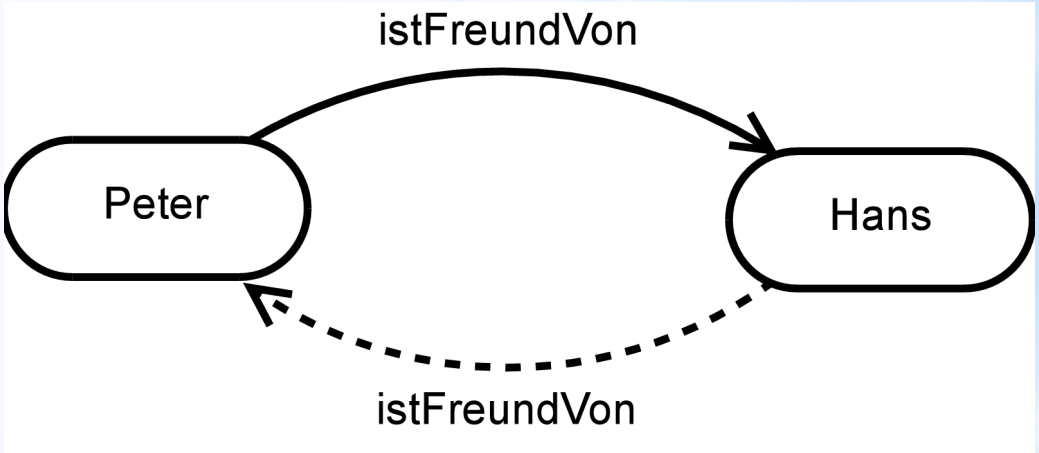
- Sind zwei Properties p_1 und p_2 als Invers zueinander gekennzeichnet, so kann aus der Existenz des Triples $A \rightarrow p_1 \rightarrow B$ das Triple $B \rightarrow p_2 \rightarrow A$ geschlossen werden

- Transitive Properties
  (Typisierung als **owl:TransitiveProperty**)



- Ist eine Property p als transitiv gekennzeichnet, so kann aus der Existenz der Triples
  $A \rightarrow p \rightarrow B$ und $B \rightarrow p \rightarrow C$ das Triple $A \rightarrow p \rightarrow C$ geschlossen werden

- Symmetrische Properties
  (Typisierung als **owl:SymmetricProperty**)

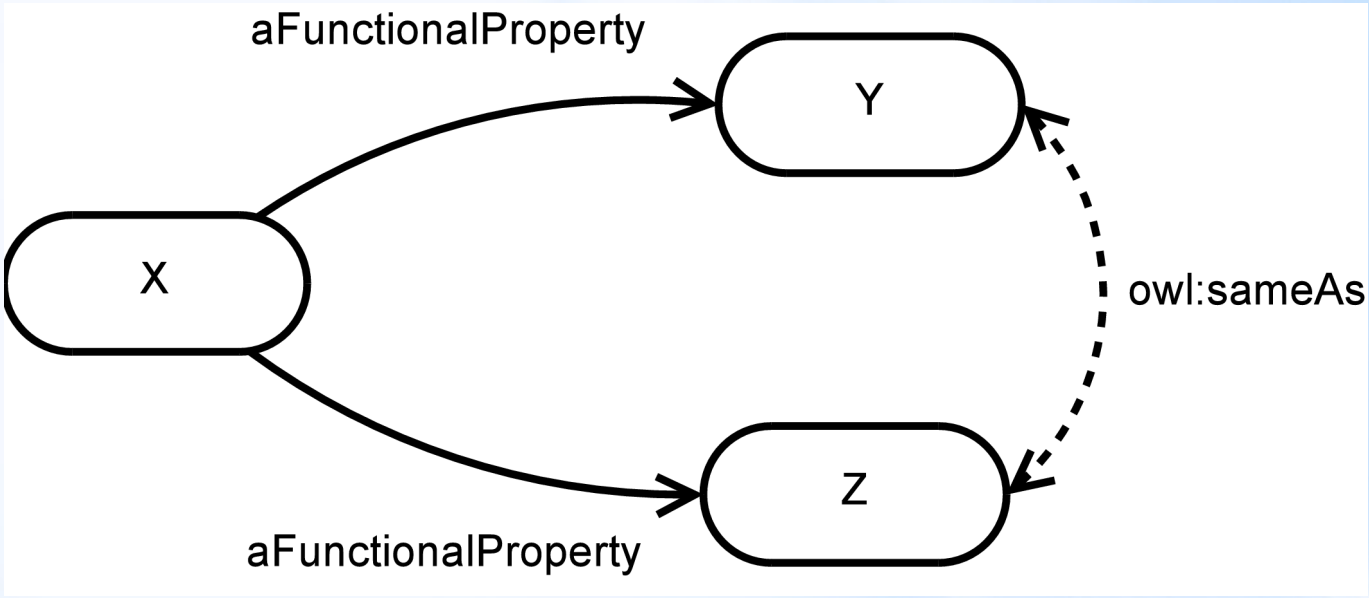


- Ist eine Property p als symmetrisch gekennzeichnet, so kann aus der Existenz des Triples

  $A \rightarrow p \rightarrow B$ das Triple $B \rightarrow p \rightarrow A$

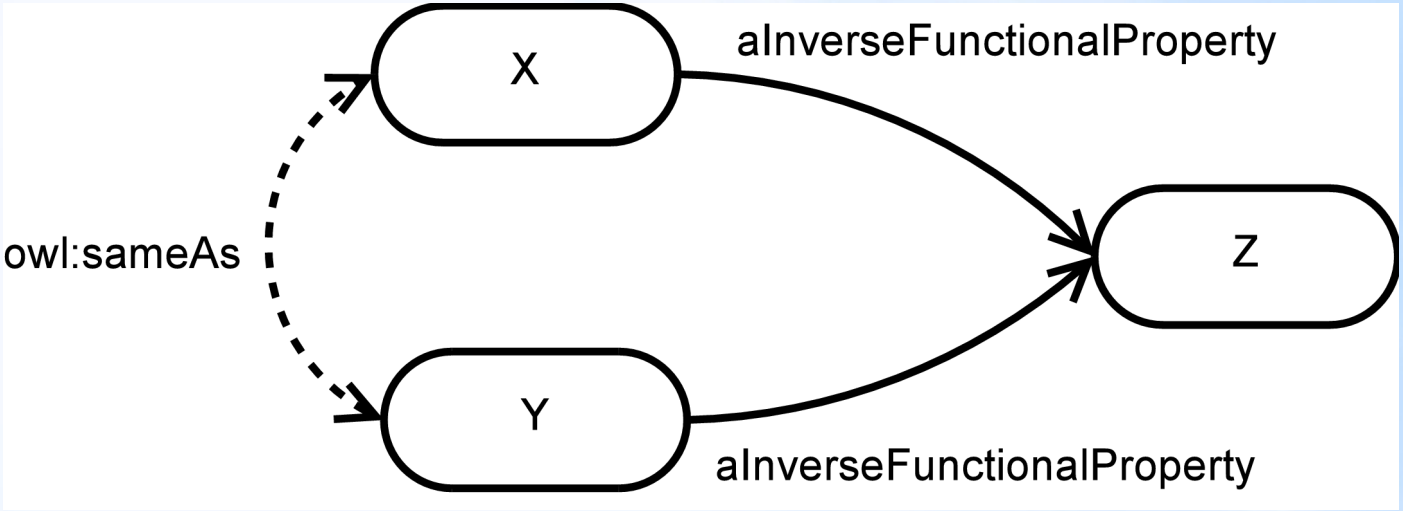
  geschlossen werden

- Funktionale Properties
  (Typisierung mit **owl:FunctionalProperty**)



- Ist eine Property p als funktional gekennzeichnet, so kann aus der Existenz der Triples
  $A \rightarrow p \rightarrow B$ und $A \rightarrow p \rightarrow C$ das Triple
  $B \rightarrow$ owl:sameAs $\rightarrow C$
  geschlossen werden

- Invers Funktionale Properties
  (Typisierung mit **owl:InverseFunctionalProperty**)



- Ist eine Property p als invers funktional gekennzeichnet, so kann aus der Existenz der Triples
  $A \rightarrow p \rightarrow C$ und $B \rightarrow p \rightarrow C$ das Triple
  $A \rightarrow$ owl:sameAs $\rightarrow B$
  geschlossen werden

- Domain und Range

```
<owl:ObjectProperty rdf:ID="Zugehoerigkeit">
   <rdfs:range rdf:resource="#Organisation" />
</owl:ObjectProperty>
```

  - Ist gleichbedeutend mit

```
<owl:Class rdf:about="&owl;Thing">
  <rdfs:subClassOf>

    <owl:Restriction>
      <owl:onProperty rdf:resource="#Zugehoerigkeit" />
      <owl:allValuesFrom rdf:resource="#Organisation" />
    </owl:Restriction>

  </rdfs:subClassOf>
</owl:Class>
```

OWL

- VORSICHT mit domain und range:

```
<owl:ObjectProperty rdf:ID="Zugehoerigkeit">
  <rdfs:range rdf:resource="#Organisation"/>
</owl:ObjectProperty>

<Zahl rdf:ID="Fuenf">
  <Zugehoerigkeit rdf:resource="#Primzahlen"/>
</Zahl>
```

- Aus obigen Fragmenten würde folgen, dass es sich bei “Primzahlen” um eine “Organisation” handelt

OWL Dialekte

OWL

- OWL Full
  - Enthält OWL DL und OWL Lite
  - Enthält als einziger OWL Dialekt ganz RDFS
  - Semantik enthält einige Elemente, die aus logischer Sicht problematisch sind
  - Unentscheidbar
  - Nur teilweise von Software unterstützt
- OWL DL
  - Enthält OWL Lite und ist Teilmenge von OWL Full
  - Entscheidbar
  - Wird von aktueller Software gut unterstützt
- OWL Lite
  - Ist Teilsprache von OWL DL
  - Entscheidbar
  - Wenig ausdrucksstark

- OWL Full
  - Uneingeschränkte Nutzung aller OWL und RDFS Sprachelemente (muss gültiges RDFS sein)
  - Schwierig durch z.B. nicht vorhandene Trennung zwischen Klassen, Rollen und Individuen. Dadurch:

    `owl:Thing` dasselbe wie **`rdfs:resource`**
    `owl:Class` dasselbe wie **`rdfs:Class`**
    `owl:DataTypeProperty` Unterklasse von **`owl:ObjectProperty`**
    `owl:ObjectProperty` dasselbe wie **`rdf:Property`**

OWL

- OWL Full

```
<owl:Class rdf:about="#Buch">
  <englischerName rdf:datatype="&xsd;string">
    book
  </englischerName>
  <franzoesischerName rdf:datatype="&xsd;string">
    livre
  </franzoesischerName>
</owl:Class>
```

- Inferenzen über derartige Konstrukte werden oft nicht benötigt

OWL

- OWL DL
  - Nur Verwendung von explizit erlaubten RDFS Sprachelementen
    - z.B. Die, die in den Beispielen verwendet wurden
    - Nicht erlaubt: rdfs:Class, rdfs:Property
  - Typentrennung: Klassen und Rollen müssen explizit deklariert werden
  - Konkrete Rollen (owl:DataTypeProperty) dürfen nicht als transitiv, symmetrisch, invers, oder invers funktional, deklariert werden
  - Zahlenrestriktionen dürfen nicht mit transitiven Rollen, deren Subrollen, oder Inversen davon verwendet werden

OWL

- OWL Lite
  - Alle Einschränkungen von OWL DL
  - Zusätzliche Einschränkungen
    - Nicht erlaubt:
      - OneOf, unionOf, complementOf, hasValue, disjointWith
    - Nur 0 und 1 als Kardinalitäten erlaubt
    - Einige Einschränkungen beim Auftreten von anonymen (komplexen) Klassen
      - z.B. Nur im Subjekt von rdfs:subClassOf

Anfragen an OWL Ontologien

- Terminologische Anfragen (Anfragen an Klassen und Rollen)
  - Klassenäquivalenz
  - Subklassenbeziehung
  - Disjunktheit von Klassen
  - Globale Konsistenz
    - Erfüllbarkeit, Widerspruchsfreiheit
  - Klassenkonsistenz
    - Eine Klasse ist inkonsistent, wenn sie äquivalent zu owl:Nothing ist, sie also keine Individuen haben kann.
    - Deutet oft auf Modellierungsfehler hin

```
<owl:Class rdf:about="#Buch">
   <owl:subClassOf rdf:resource="#Publikation"/>
   <owl:disjointWith rdf:resource="#Publikation"/>
</owl:Class>
```

- Assertionale Anfragen (Beinhaltet Individuen)
  - Instanzüberprüfung: Gehört gegebenes Individuum zu gegebener Klasse?
  - Suche nach allen Individuen, die in einer Klasse enthalten sind.
  - Werden zwei gegebene Individuen durch Rolle verknüpft?
  - Suche nach allen Individuenpaaren, die durch eine Rolle verknüpft sind.
  - ...Vorsicht: es wird nur nach „beweisbaren“ Antworten gesucht!

- OWL Sprachelemente
  - Kopf
    - rdfs:comment
    - rdfs:label
    - rdfs:seeAlso
    - rdfs:isDefinedBy
    - owl:versionInfo
    - owl:priorVersion
    - owl:backwardCompatible With
    - owl:incompatibleWith
    - owl:DeprecatedClass
    - owl:DeprecatedProperty
    - owl:imports
  - Beziehungen zwischen Individuen
    - owl:sameAs
    - owl:differentFrom
    - owl:AllDifferent (zusammen mit owl:distinctMembers)
  - Vorgeschriebene Datentypen
    - xsd:string
    - xsd:integer

OWL

- Klassenkonstruktoren und -beziehungen
  - owl:Class
  - owl:Thing
  - owl:Nothing
  - rdfs:subClassOf
  - owl:disjointWith
  - owl:equivalentClass
  - owl:intersectionOf
  - owl:unionOf
  - owl:complementOf
- Rollenrestriktionen
  - owl:allValuesFrom
  - owl:someValuesFrom
  - owl:hasValue
  - owl:cardinality
  - owl:minCardinality
  - owl:maxCardinality
  - owl:oneOf

OWL

- Rollenkonstruktoren, -beziehungen und -eigenschaften
  - owl:ObjectProperty
  - owl:DatatypeProperty
  - rdfs:subPropertyOf
  - owl:equivalentProperty
  - owl:inverseOf
  - rdfs:domain
  - rdfs:range
  - owl:TransitiveProperty
  - owl:SymmetricProperty
  - owl:FunctionalProperty
  - owl:InverseFunctionalProperty

- Open World Assumption vs Closed Word Assumption

  - Open World Assumption
    - Es können weitere Individuen existieren, sofern dies nicht explizit ausgeschlossen wird.
    - **OWL verwendet die OWA**

  - Closed World Assumption
    - Eine Wissensbasis wird als “Abgeschlossen” angenommen

  - Beispiel:

    ```
    Bob isChildOf Bill,
    Bob hasGender “Male”
    ```

    - Bei CWA würden wir nun annehmen, dass alle Kinder von Bill männlich sind
    - Bei der OWA können wir noch keine Antwort geben, da wir nicht wissen ob Bill noch weitere Kinder hat

Inferenz / Reasoning

- Inferenzprobleme
  - Globale Konsistenz der Wissensbasis — $KB \models false$?
    - Sind die enthaltenen Axiome sinnvoll?
    - Schließt sich etwas gegenseitig aus?
  - Klassenkonsistenz — $C \equiv \bot$?
    - Muss eine Klasse leer sein?
  - Klasseninklusion (Subsumption) — $C \sqsubseteq D$?
  - Klassenäquivalenz — $C \equiv D$?
  - Klassendisjunktheit — $C \sqcap D = \bot$?
  - Klassenzugehörigkeit — C(a) ?
    - Ist Individuum a in Klasse C enthalten?
  - Instanzgenerierung (Retrieval)
    - Alle bekannten Individuen zu einer Klasse finden

OWL

- Entscheidbarkeit von OWL DL
  - Entscheidbarkeit bedeutet …

    … dass zu jedem Inferenzproblem ein immer terminierenden Algorithmus existiert

  - Problem:
    - Wir müssen immer terminierende Algorithmen finden
      => Keine “naiven” Lösungen in Sicht
- Erinnerung: Wissensbasen bestehen aus 2 Teilen:
  - **TBox**: Axiome, die die Struktur der zu modellierenden Domäne beschreiben (konzeptionelles Schema):
    - HappyFather ≡ Man ⊓ ∃hasChild.Female ⊓ …
    - Elephant ⊑ Animal ⊓ Large ⊓ Grey
    - transitive(hasAncestor)
  - **ABox:** Axiome, die konkrete Situationen (Daten) beschreiben:
    - HappyFather(John)
    - hasChild(John, Mary)

- Rückführung auf Unerfüllbarkeit
  - Wir werden ein abgewandeltes Tableauverfahren verwenden
  - Tableau- und Resolutionsverfahren zeigen Unerfüllbarkeit einer Theorie

  → Rückführung der Inferenzprobleme auf das Finden von Inkonsistenzen in der Wissensbasis und damit die Unerfüllbarkeit der Wissensbasis

- Rückführung auf Unerfüllbarkeit
  - Globale Konsistenz der Wissensbasis $C \equiv \bot$?
    KB $\cup$ {C(a)} unerfüllbar (a neu)
  - Klasseninklusion $C \sqsubseteq D$?
    KB $\cup$ {C $\sqcap$ $\neg$D(a)} unerfüllbar (a neu)
  - Klassenäquivalenz $C \equiv D$?
    $C \sqsubseteq D$ und $D \sqsubseteq C$
  - Klassendisjunktheit $C \sqcap D = \bot$?
    KB $\cup$ {C $\sqcap$ D(a)} unerfüllbar (a neu)
  - Klassenzugehörigkeit C(a) ?
    KB $\cup$ {$\neg$C(a)} unerfüllbar (a neu)
  - Instanzgenerierung (Retrieval) alle C(x) finden
    - Prüfe Klassenzugehörigkeit für alle Individuen
    - Schwierig gut zu implementieren

- Tableau Transformation in NNF

  Gegeben sei eine Wissensbasis W

  - Ersetze $C \equiv D$ durch $C \sqsubseteq D$ und $D \sqsubseteq C$
  - Ersetze $C \sqsubseteq D$ durch $\neg C \sqcup D$
  - Wende die Regeln der nächsten Folie an, bis es nicht mehr geht

- Resultierende Wissensbasis NNF(W)
  - **Negationsnormalform** von W
  - Negation steht nur noch direkt vor atomaren Klassen

OWL

- Umformungsregeln
  - NNF(C) = C, falls C atomar ist
  - NNF(¬C) = ¬C, falls C atomar ist
  - NNF(¬¬C) = NNF(C)
  - NNF(C ⊔ D) = NNF(C) ⊔ NNF(D)
  - NNF(C ⊓ D) = NNF(C) ⊓ NNF(D)
  - NNF(¬(C ⊔ D)) = NNF(¬C) ⊓ NNF(¬D)
  - NNF(¬(C ⊓ D)) = NNF(¬C) ⊔ NNF(¬D)
  - NNF(∀R.C) = ∀ R.NNF(C)
  - NNF(∃R.C) = ∃ R.NNF(C)
  - NNF(¬∀R.C) = ∃R.NNF(¬C)
  - NNF(¬∃R.C) = ∀R.NNF(¬C)
- W und NNF(W) sind logisch äquivalent.

- Tableau NNF Beispiel

  $P \sqsubseteq (E \sqcap U) \sqcup \neg(\neg E \sqcap D)$

  in Negationsnormalform:

  $\neg P \sqcup (E \sqcap U) \sqcup (E \sqcap \neg D)$

Naives Tableauverfahren

- Rückführung auf Unerfüllbarkeit / Widerspruch
- Idee:
  - Gegeben: Wissensbasis W
  - Erzeugen von Konsequenzen der Form
    C(a) und ¬C(a)
    bis ein Widerspruch gefunden wird
- Beispiel

  C(a)
  $(\neg C \sqcap D)(a)$
  - 2. Formel umgewandelt: $\neg C(a) \wedge \neg D(a)$
    daraus folgt unter anderem ¬C(a)
    → Widerspruch zu C(a)

- Weiteres Beispiel zum Tableauverfahren:

  $C(a)$ $\quad \neg C \sqcup D$ $\quad \neg D(a)$

  - Ableitung von Konsequenzen:
    $C(a)$
    $\neg D(a)$
    $(\neg C \sqcup D)(a)$
  - Fallunterscheidung
    1. $\neg C(a)$
       $\rightarrow$ Widerspruch
    2. $D(a)$
       $\rightarrow$ Widerspruch
  - Teilen des Tableaus in zwei Zweige

OWL

- Tableauzweig:
  - Endliche Menge von Aussagen der Form
    - C(a), ¬C(a), R(a,b)
- Tableau
  - Endliche Menge von Tableauzweigen
- Tableauzweig ist abgeschlossen,
  - Wenn er ein Paar widersprüchlicher Aussagen zu C(a) und ¬C(a) enthält
- Tableau ist abgeschlossen,
  - Wenn jeder Zweig davon abgeschlossen ist

- Tableau Erzeugung

Auswahl	Aktion
$C(a) \in W$ (ABox)	Füge C(a) hinzu
$R(a,b) \in W$ (ABox)	Füge R(a,b) hinzu
$C \in W$ (TBox)	Füge C(a) für ein unbekanntes Individuum a hinzu
$(C \sqcap D)(a) \in A$	Füge C(a) und D(a) hinzu
$(C \sqcup D)(a) \in A$	Dupliziere den Zweig. Füge zum einen Zweig C(a) und zum anderen Zweig D(a) hinzu
$(\exists R.C)(a) \in A$	Füge R(a,b) und C(b) für neues Individuum b hinzu
$(\forall R.C)(a) \in A$	Falls $R(a,b) \in A$, so füge C(b) hinzu

  - Ist das resultierende Tableau abgeschlossen, so ist die ursprüngliche Wissensbasis unerfüllbar
  - Man wählt dabei immer nur solche Elemente aus, die auch wirklich zu neuen Elementen im Tableau führen. Ist dies nicht möglich, so terminiert der Algorithmus und die Wissensbasis ist erfüllbar.

- Tableau Beispiel:
  - Abkürzungen
    - P … Professor
    - E … Person
    - U … Universitätsangehöriger
    - D … Doktorand
  - Wissensbasis
    - $P \sqsubseteq (E \sqcap U) \sqcup (E \sqcap \neg D)$
    - Ist $P \sqsubseteq E$ logische Konsequenz ?
  - Wissensbasis (mit Anfrage) in NNF
    - $\{\neg P \sqcup (E \sqcap U) \sqcup (E \sqcap \neg D), (P \sqcup \neg E)(a)\}$

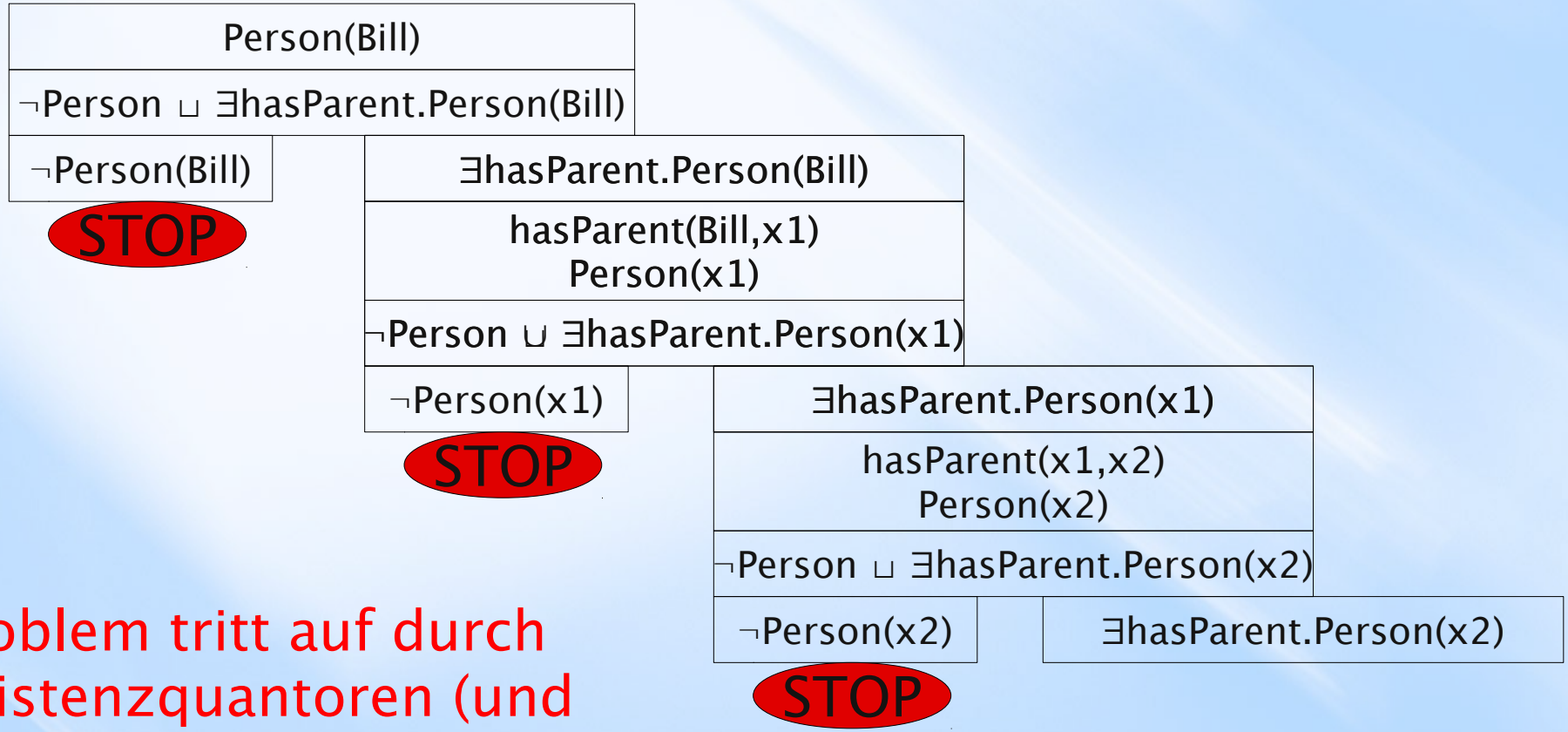
- Tableau Beispiel (2/2):
  - TBox:

    $\neg P \sqcup (E \sqcap U) \sqcup (E \sqcap \neg D)$
  - Tableau

    $(P \sqcap \neg E)(a)$ (aus Wissensbasis)
    $P(a)$
    $\neg E(a)$
    $(\neg P \sqcup (E \sqcap U) \sqcup (E \sqcap \neg D))(a)$
    $\neg P(a)$ $((E \sqcap U) \sqcup (E \sqcap \neg D))(a)$
    $(E \sqcap U)(a)$ $(E \sqcap \neg D)(a)$
    $E(a)$ $E(a)$
    $U(a)$ $\neg D(a)$

  - D.h. Wissensbasis ist nicht erfüllbar => $P \sqsubseteq E$

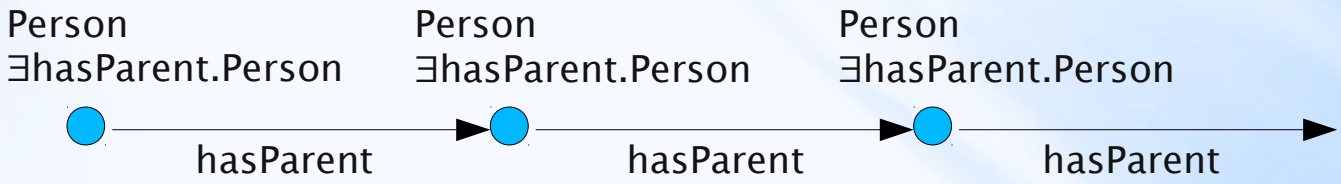
- Tableau Terminierungsproblem
  - Einziges Axiom: ¬Person ∪ ∃hasParent.Person
  - Abzuleiten: ¬Person(Bill)



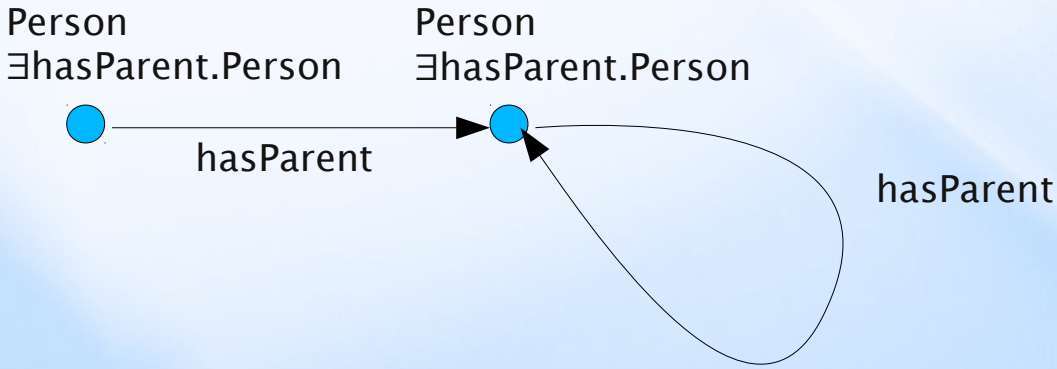
Problem tritt auf durch Existenzquantoren (und minCardinality)

usw.

- Tableau Blocking – Idee
  - Wir haben folgendes konstruiert:

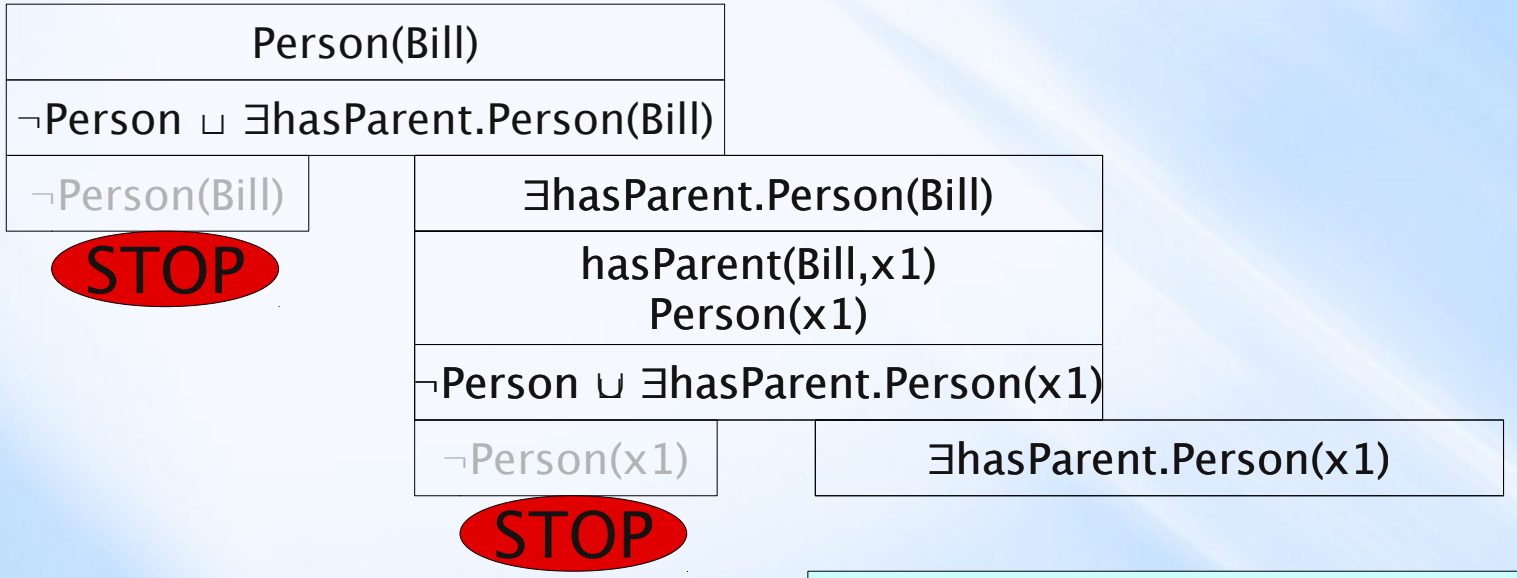


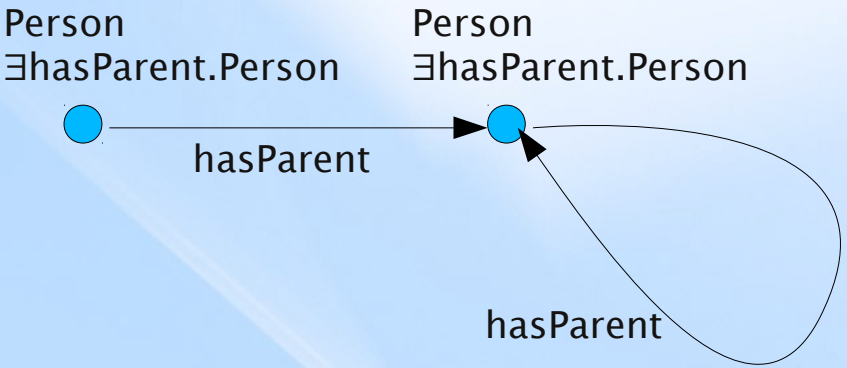
  - Folgendes wäre aber auch denkbar:



  - D.h. Widerverwendung alter Knoten!
  - Es muss natürlich formal nachgewiesen werden, dass das ausreicht

- Tableau mit Blocking
  - Einziges Axiom: ¬Person ∪ ∃hasParent.Person
  - Abzuleiten: ¬Person(Bill)





S(Bill) = {Person, Person ⊔ ∃hasParent.Person, ∃hasParent.Person}

S(X1) = {Person, Person ⊔ ∃hasParent.Person, ∃hasParent.Person}

S(x1) ⊆ s(Bill) => Bill blockiert x1

- Tableau Blocking Definition
  - Die Auswahl von $(\exists R.C)(a)$ im Tableauzweig A ist blockiert, falls es ein Individuum b gibt, so dass $\{ C \mid C(a) \in A\} \subseteq \{ C \mid C(b) \in A \}$ ist.
  - Zwei Möglichkeiten der Terminierung:
    1. Abschluss des Tableaus
       - Dann Wissensbasis unerfüllbar
    2. Keine ungeblockte Auswahl führt zu Erweiterung
       - Dann Wissensbasis erfüllbar

OWL 2

OWL

- OWL 2 als erste Weiterentwicklung des OWL-Standards
- offizieller W3C-Standard seit Oktober 2009
- logische Erweiterung: Beschreibungslogik SROIQ als Grundlage
- neue Ausdrucksmittel vor allem Rollenaxiome, qualifizierte Zahlenrestriktionen
- nicht-logische Erweiterungen: Punning, Annotationen, Datentypen, u.a.
- OWL-Profile als Ersatz von OWL Lite
- OWL 2 Full im Sinne von OWL Full definiert

OWL

- OWL 2 als “nächste Version” von OWL
  - Erweiterungen aufgrund von Praxiserfahrung mit OWL 1:
    - zusätzliche Ausdrucksstärke durch neue ontologische Axiome
    - nicht-logische Erweiterungen (Syntax, Kommentare, . . .)
    - Überarbeitung der OWL-Varianten (Lite/DL/Full)
  - Merkmale:
    - weitestgehende Kompatibilität zum existierenden OWL-Standard
    - Erhaltung der Entscheidbarkeit von OWL DL
    - Behebung von Problemen im OWL-1-Standard

OWL

- Von SHOIN(D) zu SROIQ(D)
  - OWL DL basiert auf Beschreibungslogik SHOIN (D):
    - Axiome:
      - TBox: Subklassenbeziehungen $C \sqsubseteq D$
      - RBox: Subrollenbeziehungen $R \sqsubseteq S$ (H), Inverse Rollen R^- (I), Transitivität
      - ABox: Fakten zu Klassen C(a), Rollen R(a, b), und Gleichheit $a \approx b$ bzw. $a \neq b$
    - Klassenkonstruktoren:
      - Konjunktion $C \sqcap D$, Disjunktion $C \sqcup D$, Negation $\neg C$ von Klassen
      - Rollenrestriktionen: universell $\forall R.C$ und existenziell $\exists R.C$
      - Zahlenrestriktionen (N): $\leq n\ R$ und $\geq n\ R$ (n nicht-negative Zahl)
      - Nominale (O): {a}
    - Datentypen (D)

OWL

- SHOIN unterstützt verschiedene Abox-Fakten:
  - Klassenzugehörigkeit C(a) (C komplexe Klasse),
  - Sonderfall: negierte Klassenzugehörigkeit ¬C(a) (C komplexe Klasse),
  - Gleichheit $a \approx b$,
  - Ungleichheit $a \not\approx b$
  - Rollenbeziehungen R(a, b)

OWL

- SHOIN unterstützt verschiedene Abox-Fakten:
  - Klassenzugehörigkeit C(a) (C komplexe Klasse),
  - Sonderfall: negierte Klassenzugehörigkeit ¬C(a) (C komplexe Klasse),
  - Gleichheit $a \approx b$,
  - Ungleichheit $a \not\approx b$
  - Rollenbeziehungen R(a, b)
  - Negierte Rollenbeziehungen?

OWL

- SHOIN unterstützt verschiedene Abox-Fakten:
  - Klassenzugehörigkeit C(a) (C komplexe Klasse),
  - Sonderfall: negierte Klassenzugehörigkeit ¬C(a) (C komplexe Klasse),
  - Gleichheit $a \approx b$,
  - Ungleichheit $a \not\approx b$
  - Rollenbeziehungen R(a, b)
  - Negierte Rollenbeziehungen?

→ SROIQ erlaubt auch negierte Rollen in der Abox: ¬R(a,b)

OWL

- SHOIN unterstützt nur einfache Zahlenrestriktionen (N):

$$\text{Person} \sqcap \geq 3\ \text{hatKind}$$

„Klasse aller Personen mit 3 oder mehr Kindern.“

OWL

- SHOIN unterstützt nur einfache Zahlenrestriktionen (N):

$$\text{Person} \sqcap \geq 3\ \text{hatKind}$$

„Klasse aller Personen mit 3 oder mehr Kindern.“

- SROIQ erlaubt auch qualifizierte Zahlenrestriktionen (Q):

$$\text{Person} \sqcap \geq 3\ \text{hatKind}.(\text{Frau} \sqcap \text{Professor})$$

„Klasse aller Personen mit 3 oder mehr Töchtern, die Professoren sind.“

OWL

- Das Konzept SELF

- Modellierungsaufgabe:
  „Jeder Mensch kennt sich selbst.“

OWL

- Das Konzept SELF

- Modellierungsaufgabe:
  „Jeder Mensch kennt sich selbst.“

- SHOIN :
  - kennt(tom, tom) kennt(tina, tina) kennt(udo, udo)
  - nicht allgemein anwendbar

OWL

- Das Konzept SELF
- Modellierungsaufgabe:
  „Jeder Mensch kennt sich selbst.“
- SHOIN :
  - kennt(tom, tom) kennt(tina, tina) kennt(udo, udo)
  - nicht allgemein anwendbar
- SROIQ: spezieller Ausdruck Self
  - Mensch $\sqsubseteq \exists$kennt.Self
  - ...

OWL

- Rollenaxiome und die universelle Rolle
- SROIQ bietet zusätzliche Aussagen über Rollen:
  - Tra(R): R ist transitiv (definiert wie in SHOIN)
    - Beispiel: Tra(liegtIn)
  - Sym(R): R ist symmetrisch (definiert wie in SHOIN)
    - Beispiel: Sym(verwandtMit)
  - Ref(R): R ist reflexiv, $(x, x) \in R^I$ für alle Domänenindividuen x
    - Wenig sinnvoll, da immer auf ganze Domänene bezogen!
  - Irr(R): R ist irreflexiv, $(x, x) \notin R^I$ für alle Domänenindividuen x
    - Beispiel: Irr(hatKind)
  - Dis(R, S): R und S sind disjunkt, $(x, y) \notin R^I \cap S^I$ für alle x, y
    - Beispiel: Dis(hatVater, hatSohn)
  - Universelle Rolle U: $(x, y) \in U^I$ für alle x, y
    - Beispiel: $\top \sqsubseteq \leq 7000000000$ U.Menschen (nicht empfohlen!)
    - U ist vor allem als Gegenstück zu $\top$ sinnvoll, z.B. als Wurzel der Rollenhierarchie in grafischen Editoren

OWL

- „Die Freunde meiner Freunde sind auch meine Freunde.“
  - Kann in SHOIN ausgedrückt werden: hatFreund ist transitiv.
- „Die Feinde meiner Freunde sind auch meine Feinde.“
  - Kann nicht in SHOIN ausgedrückt werden!

Rolleninklusion

- RBox-Ausdrücke der Form $R_1 \circ R_2 \circ \ldots \circ R_n \sqsubseteq S$,
  Beispiel: hatFreund ◦ hatFeind ⊑ hatFeind
- Semantik: wenn $(x_0 , x_1) \in R_1 , (x_1 , x_2) \in R_2 , \ldots , (x_n-1 , x_n) \in R_n$, dann gilt auch $(x_0 , x_n) \in S$
- Beispiel: wenn $(x, y) \in \text{hatFreund}^I$ und $(y, z) \in \text{hatFeind}^I$, dann gilt auch $(x, z) \in \text{hatFeind}^I$

- Weitere Beispiele:
  - teilVon ◦ gehört ⊑ gehört
  - hatBruder ◦ hatKind ⊑ istOnkelVon

- Problem: OWL mit Rolleninklusionen ist unentscheidbar
  - Kann man daran was ändern?
  - Rboxen sind wie Grammatiken für kontextfreie formale Sprachen
  - Überschneidungen von kontextfreien Sprachen problematisch
    → Einschränkung auf reguläre Sprachen

Reguläre Rboxen

Rollennamen werden mit $\prec$ geordnet (strenge totale Ordnung).
Jede RBox-Inklusion muss eine der folgenden Formen haben:

- $R \circ R \sqsubseteq R$
- $R^- \sqsubseteq R$
- $S1 \circ S2 \circ \ldots \circ Sn \sqsubseteq R$
- $R \circ S1 \circ S2 \circ \ldots \circ Sn \sqsubseteq R$
- $S1 \circ S2 \circ \ldots \circ Sn \circ R \sqsubseteq R$

Dabei gilt: $Si \prec R$ für alle $i = 1, 2, \ldots, n$.

RBox ist regulär, wenn es so eine Ordnung $\prec$ gibt.

OWL

- Beispiel:

  $R \circ S \sqsubseteq R \; S \circ S \sqsubseteq S \qquad R \circ S \circ R \sqsubseteq T$

  - ist regulär mit Ordnung $S \prec R \prec T$

- Beispiel:

  $R \circ T \circ S \sqsubseteq T$

  - ist nicht regulär (unzulässige Inklusions-Form)

- Beispiel:

  $R \circ S \sqsubseteq S \qquad S \circ R \sqsubseteq R$

  - ist nicht regulär (keine gültige Ordnung möglich)

- Beschränkung einfacher Rollen
  - Einfache Rollen in SHOIN = Rollen ohne transitive Unterrollen
  - In SROIQ: Beachtung der Rolleninklusionen nötig!

  - Einfache Rollen sind alle Rollen . . .
    - die nicht auf der rechten Seite einer Rolleninklusion vorkommen,
    - die Inverse von anderen einfachen Rollen sind,
    - die nur auf der rechten Seite von Rolleninklusionen $R \sqsubseteq S$ vorkommen, bei denen links einzelne einfache Rollen stehen.

  (Achtung: induktive Definition)

  → nicht-einfach sind Rollen, die direkt oder indirekt von Rollenverkettungen ($\circ$) abhängen

  - Warum ist das wichtig?

    Ausdrücke $\leq n\, R.C$, $\geq n\, R.C$, $Irr(R)$, $Dis(R, S)$, $\exists R.Self$, $\neg R(a, b)$

    nur für einfache Rollen R und S erlaubt!

    (Grund: Sicherstellung von Entscheidbarkeit)

- **Zusammenfassung strukturelle Beschränkungen**
  - **Regularität**: Einschränkung des möglichen Zusammenpiels von RBox-Axiomen
  - **Einfachheit von Rollen**: Einschränkung der Verwendbarkeit von Rollen in Zahlenrestriktionen
  - Einschränkungen der Gesamtstruktur einer Wissensbasis, bei der alle Axiome betrachtet werden müssen
  - Vereinigung mehrerer SROIQ Wissensbasen kann diese Einschränkungen verletzen, auch wenn die einzelnen Wissensbasen sie erfüllen!

OWL – Überblick über SROIQ

- Klassenausdrücke
  - Klassennamen A, B
  - Konjunktion $C \sqcap D$
  - Disjunktion $C \sqcup D$
  - Negation $\neg C$
  - Exist. Rollenrestr. $\exists R.C$
  - Univ. Rollenrestr. $\forall R.C$
  - Self $\exists S.Self$
  - Größer-als $\geq n\ S.C$
  - Kleiner-als $\leq n\ S.C$
  - Nominale $\{a\}$
- Rollen
  - Rollennamen R, S, T
  - einfache Rollen S, T
  - Inverse Rollen R^-
  - Universelle Rolle U
- TBox (Klassenaxiome)
  - Inklusion $C \sqsubseteq D$
  - Äquivalenz $C \equiv D$
- RBox (Rollenaxiome)
  - Inklusion $R1 \sqsubseteq R2$
  - Allgemeine Inkl. $R1 \circ \ldots \circ Rn \sqsubseteq R$
  - Transitivität Tra(R)
  - Symmetrie Sym(R)
  - Reflexivität Ref(R)
  - Irreflexivität Irr(S)
  - Disjunktheit Dis(S, T)
- ABox (Fakten)
  - Klassenzugehörigkeit C(a)
  - Rollenbeziehung R(a, b)
  - Neg. Rollenbeziehung $\neg$S(a, b)
  - Gleichheit $a \approx b$
  - Ungleichheit $a \not\approx b$

OWL

- Inferenz mit SROIQ
  - Rückblick: SHOIN (OWL DL) ist sehr komplex (NE XP T IME)
  - Wie komplex ist SROIQ?
  - Beobachtung: einige Ausdrucksmittel sind nicht wirklich nötig
    - Tra(R) durch $R \circ R \sqsubseteq R$ ausdrückbar
    - Sym(R) durch $R^- \sqsubseteq R$ ausdrückbar
    - Irr(S) durch $\top \sqsubseteq \neg \exists S.\mathrm{Self}$ ausdrückbar
    - Universelle Rolle darstellbar mit Hilfsaxiomen
    - $\top \sqsubseteq \exists R.\{a\}$, $R \circ R^- \sqsubseteq U$ (hier sind a und R neue Hilfssymbole)
    - ABox durch Nominale darstellbar, z.B. R(a, b) durch $\{a\} \sqsubseteq \exists R.\{b\}$
  - Qualifizierte Zahlenrestriktionen kaum problematisch
- Hauptproblem: Rollenaxiome (RBox)

- Wie geht man mit RBoxen um?
  - RBox-Regeln ähneln formalen Grammatiken
  - jede Rolle R definiert eine reguläre Sprache:
    - die Sprache der Rollen-Ketten, aus denen R folgt
  - reguläre Sprachen ≡ reguläre Ausdrücke ≡ endliche Automaten
- Ansatz: Tableauverfahren werden mit „RBox-Automaten“ erweitert

Tableauverfahren von SROIQ verfügbar:

- SROIQ ist entscheidbar.
  - Tableau-Verfahren ungeeignet für enge Komplexitätsabschätzungen
  - Komplexitätsresultat (2008): SROIQ ist N2ExpTime-vollständig!
  - Aber: Tableau-Algorithmus hat gute Anpassungseigenschaften:
    - ungenutze Merkmale belasten die Abarbeitung kaum („pay as you go“)

OWL 2 DL

OWL

- SROIQ ist „nur“ logische Grundlage von OWL 2 DL
- Weitere nicht-logische Aspekte:
  - Syntax (Erweiterung nötig)
  - Datentypdeklaration und Datentypfunktionen, neue Datentypen
  - Metamodellierung: „Punning“
  - Kommentarfunktionen und ontologische Metadaten
  - Invers-funktionale konkrete Rollen (DatatypeProperties): „Simple Keys“
  - Mechanismen zu Ontologieimport
  - …
- Hier: Übersicht einiger Kernaspekte

OWL

- Metamodellierung
  - Spezifikation ontologischen Wissens über einzelne Elemente der Ontologie (einschließlich Klassen, Rollen, Axiome).
- Beispiele:
  - „Die Klasse Person wurde am 3.1.2009 von MarkusK angelegt.“
  - „Für die Klasse Stadt wird die Property Einwohnerzahl empfohlen.“
  - „Die Aussage ‚Dresden wurde 1206 gegründet‘ wurde maschinell ermittelt mit einer Sicherheit von 85%.“
  - (Vergleich auch Reifikation in RDF Schema)

OWL

- Metamodellierung in ausdrucksstarken Logiken ist gefährlich und teuer!

- OWL 2 unterstützt zurzeit einfachste Form von Metamodellierung:
  - Punning
    - Bezeichner für Klassen, Rollen, Individuen müssen nicht disjunkt sein (Ausnahme: ObjectPropertys und DataPropertys)
    - keine logische Beziehung zwischen Klasse, Individuum und Rolle gleichen Namens
    - Beziehung nur relevant für pragmatische Interpretation

- Beispiel:
  - Person(Sebastian)
  - klasseErstelltVon(Person, Markus)

OWL

- Punning unterstützt einfache Metadaten mit (schwacher) semantischer Bedeutung
- Wie kann man rein „syntaktische“ Kommentare zu einer Ontologie machen?
  - Kommentare in XML-Dateien: `<!-- Kommentar -->`
    - kein Bezug auf OWL-Axiome dieser Datei
  - nicht-logische Annotationen in OWL 1:
    - owl:AnnotationProperty
    - fest verknüpft mit (semantischem) ontologischem Element, kein syntaktischer Bezug
- OWL 2 verändert die Bedeutung von Annotationen: keine semantische Interaktion, aber struktureller Teil von OWL-Ontologien.
- Außerdem: Annotationen ganzer Axiome möglich, nicht nur von Individuen

OWL

- OWL 2 kann in verschiedenen Syntaktischen Formen ausgedrückt werden:
  - Funktionale Syntax: ersetzt „Abstrakte Syntax“ von OWL 1
  - RDF-Syntax: Erweiterung der bestehenden OWL/RDF-Abbildung
  - XML-Syntax: Eigenständige XML-Serialisierung
  - Manchester Syntax: menschenlesbare Syntax, besonders für Ontologieeditoren
- funktionale Syntax einfacher zu definieren (keine RDF-Beschränkungen), kompakter
- RDF-Syntax für Abwärtskompatibilität wichtig

OWL 2 Lite

OWL

- OWL Lite in OWL 1 als Fehlschlag:
  - beinahe so komplex wie OWL DL
  - komplizierte Syntax gibt keinen direkten Zugang zu wahrer Ausdrucksstärke
  - Verwendung in Ontologien heute praktisch nur „zufällig“, nicht bewusst
- Ursprüngliches Ziel:
  - einfach und effizient implementierbarer Teil von OWL
- neuer Ansatz in OWL 2:
  - mehrere einfache Sprachprofile
  - OWL EL, QL, RL

OWL

- OWL EL
  - OWL-Profil basierend auf Beschreibungslogik EL++ :
  - Beschreibungslogik EL++
    - Konzepte nur mit Konjunktion $C \sqcap D$, Existenz $\exists R.C$, $\top$ und $\bot$
    - Nominale, eingeschränke Property-Ranges
    - allgemeine Rolleninklusionen (RBox), Transitivität
  - Vorteile:
    - polynomielle Komplexität
    - schnelle Implementierungen verfügbar
    - unterstützt praktisch relevante Ontologien (z.B. SNOMED-CT)

- OWL QL
  - OWL-Profil basierend auf Beschreibungslogik DL Lite:
  - Beschreibungslogik DL Lite
    - Oberklassen (rechte Seite von $\sqsubseteq$): $\sqcap$, $\neg$, $\exists R.C$
    - Unterklassen (linke Seite von $\sqsubseteq$): $\exists R.\top$
    - Inverse Rollen, einfache Rollenhierarchien
    - ABox wie in SROIQ
  - Vorteile:
    - sub-polynomielle Komplexität (verwandt mit relationalen Datenbanken)
    - schnelle Implementierungen verfügbar
    - geeignet für besonders große Datenmengen

OWL

- OWL RL
  - OWL-Profil basierend auf Horn-Regel-Fragment von OWL 2:
  - Horn-Regel-Fragment von OWL 2
    - Oberklassen (rechte Seite von $\sqsubseteq$): $\sqcap$, $\exists R.\{a\}$, $\forall R.C$, $\leq 1R.C$
    - Unterklassen (linke Seite von $\sqsubseteq$): $\sqcap$, $\sqcup$, $\exists R.C$, $\exists R.\{a\}$
    - Keine negierten Fakten, keine Reflexivität, sonst alle Rbox-Features
  - Vorteile:
    - polynomielle Komplexität
    - relativ einfache Implementierung (OWL-Axiome als Regeln)
    - verwandt mit Regelsprachen

OWL 2 Full

OWL

- Erweiterung von OWL Full um neue OWL-2-Konstrukte
- Semantik (großteils) als Erweiterung der OWL-Full-Semantik
- gedacht eher als konzeptionelle Modellierungssprache, zurzeit wenig Softwareunterstützung für automatische Ableitungen
- logische Konsistenz der Spezifikation weiter offen (wie bei OWL Full)

- viele OWL-Full-Ontologien nunmehr auch als OWL 2 DL interpretierbar (siehe z.B. Punning)

Noch Fragen ?

OWL

- Literatur:
  - Buch “Semantic Web Grundlagen”, Springer Verlag 2008
    Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph, York Sure
    ISBN: 978-3-540-33993-9
  - W3C: Web Ontology Language Overview
    http://www.w3.org/TR/owl-features/
  - W3C: Web Ontology Language Reference
    http://www.w3.org/TR/owl-ref/
  - W3C: Web Ontology Language Guide
    http://www.w3.org/TR/owl-guide/
  - W3C: Web Ontology Language Semantics and Abstract Syntax
    http://www.w3.org/TR/owl-semantics/
  - W3C: OWL 2 Web Ontology Language Document Overview
    http://www.w3.org/TR/owl2-overview/
  - Deutsche Übersetzungen von W3C Dokumenten
    http://www.w3.org/2005/11/Translations/Lists/ListLang-de.html